

Asyhadu alla ilaaha illallah  
wa asyhadu anna muhammadar rasuulullah ...

Allaahu akbar ...

Inna shalaati wa nusuuki wa mahyaaya wa mamaatii lillaahi  
rabbil 'aalamiin ...

Innalillaahi wa inna ilaihi roji'uun ...

## STATUS OF THESIS

Title of thesis

Recovery Model for Survivable System through Critical Service Resource Reconfiguration

I IRVING VITRA PAPUTUNGAN

hereby allow my thesis to be placed at the Information Resource Center (IRC) of Universiti Teknologi PETRONAS (UTP) with the following conditions:

1. The thesis becomes the property of UTP.
2. The IRC of UTP may make copies of the thesis for academic purposes only.
3. This thesis is classified as

☐

Confidential

☒

Non-confidential

If this thesis is confidential, please state the reason:

The contents of the thesis will remain confidential for \_\_\_\_ – \_\_\_\_ years.

Remarks on disclosure:

Endorsed by



IRVING VITRA PAPUTUNGAN

Perum Sendang Adi Permai A8  
Mlati Sleman 55285  
Yogyakarta, Indonesia

Date: 26/5/2008



AZWEEN ABDULLAH

Universiti Teknologi  
PETRONAS  
Malaysia

Date: 2/6/08

UNIVERSITI TEKNOLOGI PETRONAS

Approval by Supervisor (s)

The undersigned certify that they have read, and recommend to The Postgraduate Studies Programme for acceptance, a thesis entitled "**Recovery Model for Survivable System through Critical Service Resource Reconfiguration**" submitted by (**Irving Vitra Paputungan**) for the fulfilment of the requirements for the degree of Master of Science in Information Technology.

2/6/08  
Date

Signature :



Dr Azween Bin Abdullah  
Senior Lecturer  
Information Technology/Information Systems  
Universiti Teknologi PETRONAS  
31750 Tronoh  
Perak Darul Ridzuan

Main Supervisor :

Dr. Azween Bin Abdullah

Date :

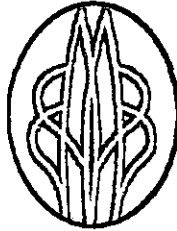
2/6/08

Co-Supervisor :

**TITLE PAGE**

**UNIVERSITI TEKNOLOGI PETRONAS**

**Recovery Model for Survivable System through Resource Reconfiguration**



**UNIVERSITI  
TEKNOLOGI  
PETRONAS**  
*engineering futures*

by

**Irving Vitra Paputungan**

**A THESIS**

**SUBMITTED TO THE POSTGRADUATE STUDIES PROGRAMME**

**AS A REQUIREMENT FOR THE**

**DEGREE OF MASTER OF SCIENCE**

**INFORMATION TECHNOLOGY**

**BANDAR SERI ISKANDAR,**

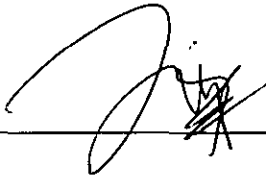
**PERAK**

**APRIL 2008**

## DECLARATION

I hereby declare that the thesis is based on my original work except for quotations and citations which have been duly acknowledge. I also declare that it has not been previously or concurrently submitted for any other degree at UTP or other institutions.

Signature :



Name :

Irving Vitra Paputungan

Date :

26/5/2008

## ABSTRACT

A survivable system is able to fulfil its mission in a timely manner, in the presence of attacks, failures, or accidents. It has been realized that it is not always possible to anticipate every type of attack or failure or accident in a system, and to predict and protect against those threats. Consequently, recovering back from any damage caused by threats becomes an important attention to be taken into account. This research proposed another recovery model to enhance system survivability. The model focuses on how to preserve the system and resume its critical service while incident occurs by reconfiguring the damaged critical service resources based on available resources without affecting the stability and functioning of the system. There are three critical requisite conditions in this recovery model: the number of pre-empted non-critical service resources, the response time of resource allocation, and the cost of reconfiguration, which are used in some scenarios to find and re-allocate the available resource for the reconfiguration. A brief specifications using Z language are also explored as a preliminary proof before the implementation. . To validate the viability of the approach, two instance cases studies of real-time system, delivery units of post office and computer system of a company, are provided in ensuring the durative running of critical service. The adoption of fault-tolerance and survivability using redundancy re-allocation in this recovery model is discussed from a new perspective. Compared to the closest work done by other researchers, it is shown that the model can solve not only single fault and can reconfigure the damage resource with minimum disruption to other services.

## **ABSTRAK**

Suatu kebolehtahanan sistem mampu mencapai misi dalam masa yang ditetapkan apabila berlaku serangan, kegagalan dan kemalangan. Hakikatnya, segala ancaman seperti serangan, kegagalan dan kemalangan selalunya tidak mungkin dapat dijangka. Oleh sebab itu, penyelesaian terhadap sebarang kemusnahan yang disebabkan oleh pelbagai ancaman ini menjadi satu keperluan yang penting untuk diambil perhatian. Kajian ini mencadangkan satu model penyelesaian untuk meningkatkan kebolehtahanan sesuatu sistem. Model ini memfokuskan kendala bagaimana mempertahankan sistem dan meneruskan perkhidmatan yang kritikal semasa berlakunya insiden dengan cara mengkonfigurasi semula sumber kemusnahan perkhidmatan kritikal berdasarkan sumber yang ada tanpa memberi kesan terhadap kestabilan sistem. Terdapat tiga keadaan kritikal yang diperlukan untuk menghasilkan model penyelesaian iaitu bilangan perkhidmatan kritikal, masa tindak balas sumber dan, kos pengkonfigurasi semula dimana semua ini telah digunakan dalam beberapa senario untuk mengatur sumber yang sedia ada untuk pengkonfigurasi semula. Spesifikasi ringkas menggunakan bahasa Z telah dikaji untuk menghasilkan pembuktian awal sebelum dilaksanakan. Pengambilan “fault-tolerance” menggunakan penstrukturan semula yang lewah dalam model penyelesaian ini telah dibincangkan dari perspektif baru. Berbanding dengan penyelidikan-penyelidikan yang telah dilakukan oleh para penyelidik, dapat disimpulkan bahawa model ini bukan sahaja boleh menyelesaikan satu kesalahan tetapi boleh mengenal pasti sumber kerosakan dengan gangguan yang minimum keatas perkhidmatan lain.

## ACKNOWLEDGEMENTS

First of all, I cannot thank enough to my beloved wife, adored son, and my big family for always supporting me in my personal and academic endeavours. Their relentless encouragement has enabled me to persevere even in times of duress.

I would like to thank my supervisor Dr. Azween Abdullah for his supervision and guidance throughout the whole work with this thesis. I greatly appreciate his encouragement to go ahead with my work especially when lacking of self confidence and expertise.

I would also like to thank Dr. Fadzil Hassan, Mr. Low Tan Jung and Ms. Nor Shuhani for their helpful comments, comprehension, and reviewing my research papers even with very short notices. I also want to thank Dr. Ahmad Kamil Mahmood, the head of Computer and Information Sciences Department of UTP, for his support.

My sincere thanks to all the administrative staff: Mr. Azful, Kak Norma, Kak Wahida, Kak Aida, Mr. Fadil Ariff and the rest of postgraduate office staff for their assistance during my time in UTP.

I am grateful to Mr. Zainudin Zukhri for his patient being my guru, Dicky for the copy and paste effort, and all Indonesian in UTP, you are my inspiration.

Last but not least, I would like to address my appreciation to my former colleagues, faculty, and staff of Islamic University of Indonesia. Thank you for all the experiences and knowledge. Without them, I would not have the opportunity to pursue my degree in UTP.



## TABLE OF CONTENTS

STATUS OF THESIS .....	i
TITLE PAGE.....	iii
DECLARATION.....	iv
ABSTRACT .....	v
ABSTRAK .....	vi
ACKNOWLEDGEMENTS .....	vii
TABLE OF CONTENTS .....	viii
LIST OF FIGURES .....	x
LIST OF TABLES .....	xii
NOMENCLATURE .....	xiii
CHAPTER 1 : INTRODUCTION.....	1
1.1 Background.....	1
1.2 Problem Statement.....	4
1.3 Motivation .....	4
1.4 Objective.....	5
1.5 Methodology.....	5
1.6 Thesis Overview .....	7
CHAPTER 2 : LITERATURE REVIEW.....	8
2.1 Related Work.....	8
2.2 Survivability .....	11
2.3 Recovery and Fault Tolerance.....	12
2.4 Redundancy and Resource Reconfiguration .....	14
2.5 Real Time System.....	15
2.6 Summary.....	16
CHAPTER 3 : MODEL DEVELOPMENT .....	17
3.1 Problem Definition .....	17
3.2 System Transition Diagram .....	19

3.3 Recovery Model .....	20
3.3.1 Recovery Action .....	20
3.3.2 Resource Balancing .....	24
3.3.3 Self Reconfiguration .....	25
3.3.4 Diagnosis and Analysis Process .....	26
3.3.5 Allocation Process .....	27
3.3.6 General Interconnected Network Recovery Model .....	39
3.3.7 Recovery Algorithm .....	44
3.4 Summary .....	45
CHAPTER 4 : SPECIFICATION AND SIMULATION MODEL .....	46
4.1 Z Notation .....	46
4.2 Formal Z Notation .....	47
4.3 Simulation Model .....	49
4.3.1 Single Fault Case Study .....	52
4.3.2 Multiple Sequential Faults Case Study .....	56
4.3.3 Multiple Concurrent Faults Case Study .....	60
4.4 Result and Analysis .....	67
4.5 Summary .....	68
CHAPTER 5 : CONCLUSION .....	69
CHAPTER 6 : FUTURE WORK .....	71
REFERENCES .....	73
APPENDIX A: The Allocation Flowchart .....	79
APPENDIX B: Simulation Program Code .....	80

## LIST OF FIGURES

Figure 3-1: Real-Time System Structure.....	18
Figure 3-2: State Transition Diagram.....	19
Figure 3-3: Modules in the Recovery Action .....	21
Figure 3-4: Queue Abstract Structures .....	21
Figure 3-5: Resource Availability and Reallocation .....	24
Figure 3-6: Concurrent Allocation Queue .....	26
Figure 3-7: Graph of The Services .....	27
Figure 3-8: Possible schemes for Scenario 3.3.....	33
Figure 3-9: Possible schemes for Scenario 4.3.....	37
Figure 3-10: the Interconnected Network.....	39
Figure 3-11: Interconnected Network Recovery for Case 2.....	41
Figure 3-12 : Interconnected Network Recovery for Case 3.....	42
Figure 3-13: Recovery Algorithm .....	44
Figure 4-1 : Resource Quantity User Interface.....	50
Figure 4-2 : Resource response time User Interface .....	51
Figure 4-3 : Resource cost User Interface .....	51
Figure 4-4 : Post Office Mail-Men Case Study.....	52
Figure 4-5 : Damage on one Critical Post Office .....	53
Figure 4-6 : Diagnose Result for Single Fault.....	53
Figure 4-7 : Analysis Result for Single Fault .....	54
Figure 4-8 : Allocation Result for Single Fault .....	54
Figure 4-9 : Feasible Recovery Scheme for Single Fault.....	55
Figure 4-10 : Multiple Sequential Faults Case .....	56
Figure 4-11 : Sequential Faults Diagnosed Result .....	57
Figure 4-12 : Analysed Results for Sequential Faults .....	57
Figure 4-13 : Allocation result for sequential faults.....	58

Figure 4-14 : Feasible Scheme for sequential faults .....	58
Figure 4-15 : Computer Server Systems Network .....	60
Figure 4-16 : Initial State of the network .....	61
Figure 4-17 : Resource response time of the network .....	62
Figure 4-18 : Resource cost of the network.....	62
Figure 4-19 : Multiple Concurrent Faults.....	63
Figure 4-20 : Diagnosis result of Multiple Concurrent Faults .....	63
Figure 4-21 : Analysis result of Multiple Concurrent Faults .....	64
Figure 4-22 : Allocation result of Multiple Concurrent Faults.....	64
Figure 4-23 : Recovery Schemes of Multiple Concurrent Faults.....	67
Figure A-1: Allocation Flowchart .....	79
Figure B-1: The Initial Condition Process.....	80
Figure B-2: The Recovery Process.....	82

## LIST OF TABLES

Table 1-1: Key Properties of Survivability .....	3
Table 3-1 : Master Resource Allocation table .....	22
Table 3-2 : Critical Service Resources table .....	22
Table 3-3 : Non-Critical Service Resources table .....	23
Table 4-1 : Allocation Process for Multiple Concurrent Faults .....	66

## NOMENCLATURE

$CS$	: Critical Service
$NCS$	: Non Critical Service
$y$	: Index for Critical Service
$w$	: Index for Non Critical Service
$R$	: Critical Service Resources
$r$	: Non Critical Service Resources
$S$	: Master Controller Resources
$a$	: Index for $S$
$b$	: Index for $R$
$c$	: Index for $r$
$MQR$	: Master Required Used Resources
$MUR$	: Master Used Redundant Resources
$MUNR$	: Master Unused Redundant Resources
$CSCU$	: Critical Service Currently Used Resources
$CSR$	: Critical Service Redundant Resources
$CSD$	: Critical Service Damaged Resources
$NCSCU$	: Non Critical Service Currently Used Resources
$NCSR$	: Non Critical Service Redundant Resources
$\gamma$	: Master Controller Resource Amount
$\alpha$	: Critical Service Resource Amount
$\beta$	: Non Critical Service Resource Amount
$c$	: Resource cost (unit cost)
$t$	: Resource response time (unit time)
$C$	: Cost of Reconfiguration
$T$	: Time of Reconfiguration

$q$	: Reconfiguration Request Arrival time
$l$	: Critical Service level
$Q$	: Reconfiguration Request Queue
$N$	: Resource number
$\delta$	: Feasible Reconfiguration Scheme
$\Omega$	: System's Reconfiguration Scheme, it contains $\{\delta_1, \dots, \delta_i\}$

## CHAPTER 1 : INTRODUCTION

This chapter introduces the conducted research. First, the motivation of building a survivable system is given. The main objective of the research and the approach proposed come in the next section. Then, an outline of the remaining chapters of this thesis is described.

### 1.1 Background

At the present time, computer systems are used in distributed, often malicious, environments. In this condition of unprotected setting, system vulnerabilities invite misuse, intrusion, and other forms of attack. Despite the best effort of security practitioners, no amount of system hardening can assure that a system that is connected to an unbounded network will be invulnerable to be faulted [Ellison, 1999a] [Ellison, 1999b] [Ellison, 1999c], and it is known that malicious attacks (actions and software or hardware) are becoming more smart and feasible.

Traditional security technologies were not designed to protect systems against the evolving threats that result from the interconnection of systems that were not designed to be interconnected in the first place. These systems typically employ a hierarchical design approach, appropriate for systems with centralized administration and coordination. But it cannot provide a survivable solution in the context of unbounded systems. The discipline of system survivability and security can help ensure that such systems (medical, financial, manufacturing, telecommunications, etc) can deliver essential services and maintain essential properties such as integrity, confidentiality and performance, despite the presence of intrusion.



Unlike the traditional security policies that require central control instance or administration, survivability is intended to address unbounded network environments. Survivability is a new field of study that allow systems to survive, limit damage, recover, and operate robustly in the presence of attacks, failures, or accidents that, alone or in combination threatens the system's ability to fulfil their mission. Furthermore, survivable systems should be able to evolve and adapt to the environment where they provide services.

A key characteristic of survivable systems is their capability to deliver essential services, fulfil its mission in a timely manner and in the face of threats. In some cases, if an essential service is lost or destroyed, it can be replaced by another service that supports mission fulfilment in a different but equivalent way. To maintain that capability, survivable systems have '3R' [Ellison, 1999a] [Mead, 2000]: Resistance, Recognition and Recovery. In addition, adaptation is the fourth property. Resistance means how the system repels the attacks, i.e. by system and user authentication, encryption, firewalls. Recognition describes how the system detects the attack (including intrusion) and understanding the current state of the system, i.e. by virus scans, system monitoring configuration or network monitoring. Recovery means how the systems recover from damage as early as possible to fulfil its mission as conditions permit. Recovery is selected as the focus on this research. Adaptation or evolution is needed to reduce effectiveness of future attacks by improving system survivability based on knowledge gained from intrusions. It can be by adaptive filtering and logging.

Table 1-1: Key Properties of Survivability

Key Properties	Description	Example
Resistance to attacks	Strategies for repelling attacks	User authentication Stochastic diversity of programs
Recognition of attack and the extent of damage	Strategies for detecting intrusions and understanding the current state of the system, including evaluating the extent of damage	Recognition of intrusions usage patterns Internal integrity checking
Recovery of full and critical services after attack	Strategies for restoring compromised information or functionality, limiting the extent of damage, maintaining or, if necessary, restoring critical services within the time constraint of the mission, restoring full services as condition permit	Replication and re-initialization of data
Adaptation and evolution to reduce effectiveness of future attacks	Strategies for improving system survivability based on knowledge gained from intrusions	Incorporation of new patterns for intrusion recognition

The informal notion of a “threat” that has been used is what is referred to formally in the literature as a fault or error. The process of building a system in such a way that certain faults do not arise is fault avoidance. Building systems that are able to react in requisite way to prescribe faults is fault tolerance. Survivability requires robustness under conditions of intrusion, failure, or accident; and it includes the concept of fault tolerance. This research is on the subject of developing fault tolerance to enhance system survivability.

## 1.2 Problem Statement

Based on the introduction, there are several issues in survivability that can be summarized:

1. No system is invulnerable to fault.
2. Some faults are typically destroying the resources of critical services; making the services cannot be continued with resources that remained.
3. The need for a mechanism to ensure the system delivers the critical services despite the presence of faults.
4. Recovering the destroyed resources of critical services as soon as possible when conditions permit becomes a must to fulfil the mission of the system when incident occurs.

## 1.3 Motivation

The above issues manifest the dangers of system intrusions in interconnected environment, thus recovery of destroyed critical service to sustain mission of the system becomes important. The recovery will be those of restoring the damaged critical service resources in efficient ways as condition permit to increase survivability of the system. In fault tolerance, survivability can be deliberated as requiring very specific error recovery after a fault. If the system can tolerate the fault and recover back to good condition and ultimately keep the affected services running all along, the system will survive. This motivates the author to explore methods to improve the survivability of systems in fault tolerance perspectives.

An approach to achieve this is by: 1) To model the fault handling process, 2) Model a resource reconfiguration using redundant resources, and 3) Simulate alternative scenarios to

show different conditions of resource reconfiguration. This work focuses on developing such a recovery model for critical service resource reconfiguration.

#### 1.4 Objective

A primary objective of this research is to develop a recovery model that can be used by specially a system administrator to enhance system survivability by reconfiguring the faulted critical service resources. Along with this, some sub-objectives include:

- Construct a formal specification for the recovery model using Z formal specification language to pre-prove the proposed model.
- Develop a simulation application to demonstrate how this model works with some hypothetical data.

#### 1.5 Methodology

The goal is to model the recovery process of a system when its critical service is subjected to damage. The basic idea of full survivability is the system remains alive to a certain condition of failures occurring anywhere in the system [Ho, 2007]. For this reason, some relevant concepts in current literature will be used. Firstly, the State Transition Diagram of the system have been built, which is the simplification of Popstojanova's work [Popstojanova, 2001]. By taking into account the time factor [Lin, 2005] [Wang, 2006], cost factor [Moitra, 2000] [Park, 2004] [Wang, 2006], resource re-allocation/reconfiguration concept [Wang, 2006] [Aung, 2006], and resource redundancy concept [Knight, 1998] [Sullivan, 1999] [Aung, 2006], a different approach in fault tolerance perspective is developed to return the system back to normal condition.

The proposed model will assign resource redundancy at the outset and keep a table for each critical and non-critical service nodes. The model implements a technique of reconfiguring the destroyed resources using redundant or pre-empted resources at the fastest possible time with minimum service disruption. Since the system is referred to Real-Time system, the stability must be maintained. The stability of the entire system is based on active and accurate functioning of each and every service nodes. The system becomes unstable when at least one active service nodes becomes dysfunctional and its resources are pre-empted or denied access. It is assumed that error detection and assessment will be handled by other mechanism such as operating system. Since the objective is to recover the system when incident occurs, the system shall degrade gracefully [Park, 2005] when recovery process is running. Since most of the recovery techniques placed after the system fails to run, this is a different thought to put a process in graceful degradation mode. In this research, services and processes are used interchangeably.

To design a system that capable to make decision more quick and accurate than a human could, a new concept called autonomic computing is adopted in this research. Autonomic computing is a computational method that less real-time human intervention [Lewandowski, 2001]. In this work, some hypothetical data is proposed for the simulation model.

## 1.6 Thesis Overview

The remainder of this thesis is organized as follows:

Chapter two provides a general overview of enabling technologies used to address the proposed approach and its implementation.

Based on the issues highlighted in chapter one and two, chapter three presents the recovery model. The model and algorithms needed are extensively explained, included the queuing model for multiple faults and system.

Chapter four shows the specification of the model using Z formal language, a simulation model program is provided in this chapter as well. Some case studies are tested and explored.

Chapter five and six, draw the conclusions of the research and some recommendations for future research.

## CHAPTER 2 : LITERATURE REVIEW

This chapter elaborates some concepts related to this work and presents the state of the art and methodologies used in designing the recovery model: survivability, recovery and fault tolerance, redundancy and resource reconfiguration, and real-time system.

### 2.1 Related Work

There are different approaches for resolving survivability problem in term of modelling. Start with Sullivan (1999) who developed dynamic models of infrastructure information system based on the notion of control. The author wrote that dealing with disruptions, no matter what the cause, minimizing the loss aggregate value to users and ensuring that the system remain have to be taken into account. Moitra (2000) proposed a complete episodes simulation model for managing survivability of networked information system, including the responses of the system to attacks, to solve problem in [Sullivan, 1999]. The model addressed several detailed aspects of the attack incidents, such as the type of attack, the number of attackers, and possible correlation between the rate of incidents and the type of incidents. The return of the system to the normal state when an incident occurs is one of the accomplishments which should be stressed in the future. It also mentioned in [Zhao, 2006] who proposed a novel quantitative analysis method for network survivability based on grey analysis to assess the best affiliate degree and survival probability of every key service. The method obtained synthetically analysis for the network survivability by analyzing the changes of every key service's survivability. It stated that how to realize the real-time analysis of network survivability and recover the fault service will be the further research.

Returning back to normal condition in survivable system is explored briefly by [Jha, 2001]. Jha (2001) use a Constrained Markov Decision Process (CMDP) to form the basis of survivability analysis, which is composed of model-checking, Bayesian network analysis, probabilistic analysis and cost-benefit by presenting in a single framework of systematic method. The author mentioned a better distinction between survivability and traditional fault tolerance. Almost similar as [Jha, 2001], Koroma (2003) proposed a quantitative approach using a generalized model (Markov chain) to achieve how the system will function in the wake of failures, what will be the impact of failures on the user, and how to overcome these failures. The node connectivity is the only background of that work. McDermott (2005) presented a quantitative survivability modelling for high-consequences system based on intruder attack potential. The consideration of recovering system based on concurrent faults rather than sequential faults is mentioned on this work.

There are some studies on survive by recovery. Started by Knight (1998) who introduced a recovery concept based on fault tolerance and reconfigurable system on critical infrastructure in the face of catastrophic faults where the effects of the faults cannot be masked using available resources. Several suggestions on how a recovery related to reconfigurable process are presented in this work. It is also mentioned that for future system characteristic, more redundancy might be built since the cost of hardware continues to drop. Park (2004) introduced a hybrid recovery model that is compiled from static and dynamic recovery model. Besides providing more robust survivability services than two other models, the hybrid model addressed the system downtime drawback in static model by using redundant server buffer. It is mentioned when error happened to a server; the buffer will take over the service for a brief period until the new immunized server is initialized. And if the transition period is long, multiple buffer servers might be used. For further adaptation, multiple immunized servers are generated. However, the simplicity of the model becomes lower.



Wang (2006) developed a new method ERAS (Emergent Response Algorithm for Survivability of Critical Services) to sustain the operation of critical service of a system after attack. The algorithm takes the requirement of emergency response as background to find the feasible scheme of reconfiguration from the view of needed resources as early as possible and minimum use of pre-empted non-critical service resource number. It is a new thought and method for survivability and very effective to use. However, the algorithm is only considering single attack on critical service and reconfiguring from non-critical service resources. To reconfigure optimally, Ghiasi (2004) presented an efficient optimal algorithm for minimizing the run-time reconfiguration delay of executing an application on dynamically adaptable system. It is able to minimize the total application run-time from many classes partial reconfiguration delay. The author mentioned that the effect of number of reconfiguration is the next investigation

Aung (2006) presented a cluster recovery model to increase the survivability level of internet applications. The author used redundancy method to create a hot standby system in face with disasters. It also provides a mathematical model using semi-markov model. The model is able to reduce the time to get the users back to work, but it did not mention the integration of response time and throughput with downtime cost.

Hiltunen (2001) advocated the use of a standard fault tolerance technique - redundancy - to increase the survivability of service. The author mentioned that the fundamental idea behind using redundancy to improve survivability is with multiple resources enforcing a given attribute; the attribute should remain valid if at least one of the resources remains uncompromised.

Recovering a system is extremely complex. Error recovery planning is used to address the after failure conditions when the system is offline. By paying attention to the detail from previous works, it is possible to increase the likelihood of successful recovery and keep maintaining the faulted service to survive the system [Mead, 2000].

## 2.2 Survivability

The definition of survivability is not always consistent or even present in the current literature discussing survivability [Westmark, 2004]. The standard definition by [Ellison, 1999a] [Ellison, 1999b] [Ellison, 1999c] [Byon, 2000] [Caldera, 2000] [Knight, 2003] is used as the basis to explain what survivability is. *Survivability is the capability of a system to fulfil its mission, in a timely manner, in the presence of attacks, failures, or accidents.*

The term *system* is used typically in a large-scale network system, not limited to computer system, which includes many components (nodes) that are required to deliver services to the end user. The system environment and the critical services that the system provides are defined for this survivable network system. The system can be bounded or unbounded. The system is unbounded if all nodes that provide the critical services are not known.

*Mission* refers to a set of high-level requirements or goals and not limited to military settings. Any successful organization or project must have a vision, whether they are expressed implicitly or explicitly. Mission can be judged by the expectations of the user. Mission is related to critical services that system provides. For example, cheque clearing is a critical service of a banking system. It means the mission of a survivable banking system is to continue providing this service despite the presence of faults [Jha, 2001].

*Timeliness* is a critical factor that is typically included in the high-level requirements that define a mission especially in real-time system. For example: for a networked distributed system, a required service may be a specified response time to the end user. For a time-critical system, the description of a required service may include the maximum time allowable between user request and system response.

The terms *attack*, *failure*, and *accident* can be considered as threats or incidents or intrusions. It is something that may prevent the system from providing services to the user in the prescribed time or may prevent the system from providing the services at all. [Westmark, 2004] categorized the threat as: 1) Accidental threats: software errors, hardware errors, and human errors, 2) Intentional or malicious threats: sabotage, intrusion, or terrorist attacks, and 3) Catastrophic threats typically do not allow delivery of required service to the user, which includes acts of nature (thunderstorms, hurricanes, lightning, flood, earthquake, etc.), acts of war, and power failures.

Another term that can be added into survivability definition is *business case* [Westmark, 2004]. A business case is required for each survivability definition. There is an extra cost associated with the design, development, and operation of a survivable system. The business case is developed based on the cost/benefit analysis from which the threat is identified and required responses are specified.

To develop a survivable system, some analysis steps must be followed [Fung, 2005] [Mead, 2000]. The first step is to focus on understanding the system's goal. Any violation of this goal means a compromised system. Next is to identify the critical components and resources in the services, based on the goals and the consequences if failure happened. The third step is to identify the compromised components. Finally identify components that both prove and possible to be compromised after the previous compromised one.

### 2.3 Recovery and Fault Tolerance

In survivability terms, recovery is the ability of a system to restore compromised services within the time constraint of the mission as condition permit [Ellison, 1999b]. Recovery also contributes to a system's ability to maintain critical services during failure condition [Ellison, 1999a]. Requirements for recoverability are what most clearly distinguish

survivable systems from secure system. Traditional security leads to the design of systems that rely almost entirely on hardening for protection. Once security is breached, damage may follow with little to stand in the way. The ability of a system to react during an active intrusion is central to its capacity to survive an attack that cannot be completely repelled. Recovery is thus crucial during exploration and exploitation phases of intrusion.

One concept in recovery is fault tolerance. Fault tolerance is a mechanism that enables a system to continue operating properly in the event of failure of (or one or more faults within) some of its components [Tirtea, 2006]. If its operating quality decreases at all, the decrease is proportional to the severity of the failure, as compared to a naively-designed system in which even a small failure can cause total breakdown. Fault tolerance is sometime called graceful degradation which means that the recovery of system's capability and services will be placed in degraded mode.

Some damages may not be detected immediately, thus the damages may propagate until they are detected. Once they are detected, the recovery process must fix them one by one or simultaneously. Recovery also depends on the severity of the damage (i.e., how many resources have been affected), the recovery strategies and the remaining undamaged resources that are in place [Wang, 2006]. Damage assessment, recovery schemes for different degrees of damage severity, and redundant resources for reconfiguration can ensure the survivability by recovery.

This research separates the error detection and error recovery processes in which both are usually embedded with the single engine service [Florio, 2001].

## 2.4 Redundancy and Resource Reconfiguration

Redundancy is a key component in providing any kind of tolerance especially in survivable system [Hiltunen, 2003]. The term redundancy has its roots in both fault tolerant hardware/software and distributed systems. It generally refers to the extra resources allocated to a system that are beyond its need in normal working conditions [Wang, 2003]. In engineering, redundancy is the duplication of critical components of a system with the intention of increasing the reliability of a system. Redundancy is different from replication, which is just one type of redundancy which we are all concerned to: physical resource redundancy, beside software, information, or time redundancy. Redundancy can help the system avoid single points of vulnerability, including vulnerabilities and weaknesses in distributed security algorithms [Hiltunen, 2003].

A survivable system must be proactive as well as reactive [Wells, 2000]. Assigning a sufficient redundant resource for each service according to pre-specified policy in a system is proactive. The reactive is when the system needs redundant resource, due to resource failure of a service, it can be re-allocated successfully.

Resource reconfiguration or re-allocation plays a major role in survivable system and the proper use of redundant resources is highly important [Herzberg, 2004]. The re-allocation must be dynamic and policy based. This process is a successor of autonomic response to resolve lacking resource after failure. In [Koopman, 2003], a reconfiguration process to heal a system having suffered from fault without human intervention is considered to be self-healing by self-reconfiguration mechanism.

## 2.5 Real Time System

Real-time system [Greenwood, 2005] is any system that is both logically and temporally correct. Logical correctness means the system satisfies all functional specifications. Temporal correctness means the system is guaranteed to perform these functions within explicit time frames. Fault-tolerant systems meet the criteria as real-time systems because fault detection and fault recovery inherently have deadlines. That is, the fault must be detected within a certain period of time after it occurs, and the fault must be corrected within a certain period of time after it is detected. Fault recovery may also have an expected start time.

The notion of real-time is often interpreted to imply really fast. This interpretation is not accurate. Real-time does not necessarily mean fast and fast does not necessarily mean real-time [Greenwood, 2005]. Suppose a document must be sent from Kuala Lumpur to Jakarta, and two delivery systems are available: surface mail with a guaranteed three-day delivery time or e-mail with a guaranteed 7 minute delivery time. The e-mail delivery is in the order of magnitude faster than surface mail, but that does not necessarily mean it qualifies as a real-time delivery system. It is the required delivery deadline that ultimately establishes whether the real-time system definition has been met. For example, both systems are real-time systems if the deadline is six days because both are logically and temporally correct. However, neither one is a real-time system if the deadline is 3 minutes because neither one is temporally correct.

Real-time systems are classified as hard or soft. Hard systems have catastrophic consequences if the temporal requirements are not met up to and including complete system destruction. In fact, if the hard system is safety-critical, failure could lead to injury or even death. Conversely, soft systems only have degraded performance if the temporal requirements are not met. The classification of a fault-tolerant system, in particular, depends

on the nature of the faults and the consequences for failing to detect and correct them in a timely manner. Suppose an error results in system. If the system can survive this condition for up to 5 minutes, the fault recovery must be completed within 5 minutes to prevent further damages.

## 2.6 Summary

This chapter provides the reader with sufficient background information to understand the foundations and concepts elaborated in the rest of this thesis. The rest of the sections discuss the approach used in designing the survivability model, recovery and fault tolerance, redundancy and resource reconfiguration within the context to real-time systems.

## CHAPTER 3 : MODEL DEVELOPMENT

This chapter gives a detailed description of the recovery model and its actions. Firstly, the addressed problem is described. The recovery model is proposed to solve the problem. .

### 3.1 Problem Definition

The problem can be described as: There is a Real-Time System (RTS) which has critical services and non-critical services. Some critical service resources are destroyed by a fault. To maintain the stability and mission of the system, the system has adaptive abilities to recover by re-allocating available resources dynamically to critical service. The recovery process works under RTS circumstance, hence the duration of the process is a great concern. The duration included the response time and the usage time. This work focuses on the response time of the available resources. That is the first requisite condition. The problem is to find out how to reconfigure the critical service resources which can ensure sustainable operation of critical services. It brings along to the next requisite condition, cost. The cost of reconfiguration is calculated by the number of resources and cost of resource. It is assumed that error detection (monitoring) and damage assessment are taken care by some other mechanism such as control system architecture [Knight, 1998]. It is assumed that there will be no further error when a critical service being faulted. Figure 3-1 provides a typical example of interconnected system structure that contains a master resource controller, some critical services and some non-critical services, where the recovery could be applied. Resource reconfiguration computation will be done by the recovery engine.



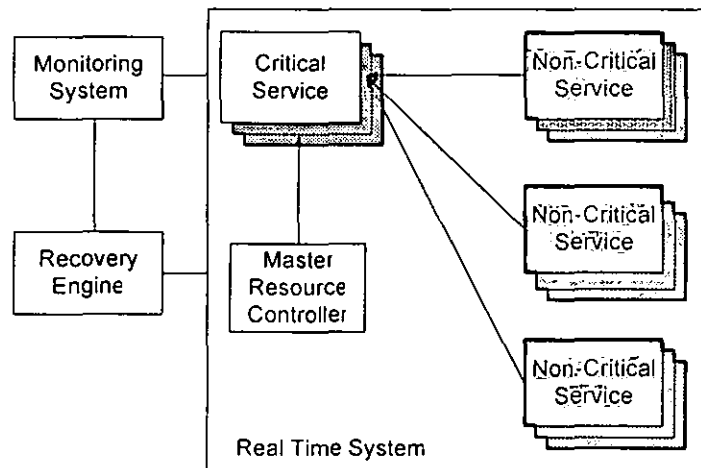


Figure 3-1: Real-Time System Structure

The critical service differs in three levels: High, Medium and Low level depending on its function at the system. The replacement resources to reconfigure the damage can be taken from 1) Redundant Resources of critical service, 2) Unused/idle Redundant Resources of the system, 3) Released/Redundant Resources of non-critical service, and 4) Pre-empted Resources of non-critical service that are currently being used. They are depicted in Figure 3-5.

To avoid instability of the system and more cost while reconfiguring, the response time of available resources should be as quick as possible and the cost of available resources should be as cheap as possible. Furthermore, the number of pre-empted non-critical service resources, if need to be utilized, should be as few as possible, this is considered the third requisite condition.

### 3.2 System Transition Diagram

Figure 3-2 depicts the state transition diagram which is used as a framework for describing the behaviour of the system. The system contains 4 states: *Good state*, *vulnerable state*, *fault state*, and *recovery state*. The system moves to vulnerable state if a user violates security policy to access a resource without authorization. Vulnerability is the property of the system, its attendant software and/or hardware, or its administrative procedures, which causes it to enter vulnerable state [Popstojanovan, 2001]. The system enters fault state when vulnerability is successfully exploited and the fault unmasked by simple fault tolerance. In the next state, recovery state, the system will be recovered. To limit the damage and protect the system from Denial of Service while maintaining the critical services, it sets into graceful degradation mode. Critical services are defined as the functions of the system that must be maintained to meet the system requirements even when the failures occurred [Ellison, 1999a]. In order to survive the critical service, it is critically assumed the recovery process will always be successful; hence there is no fail state.

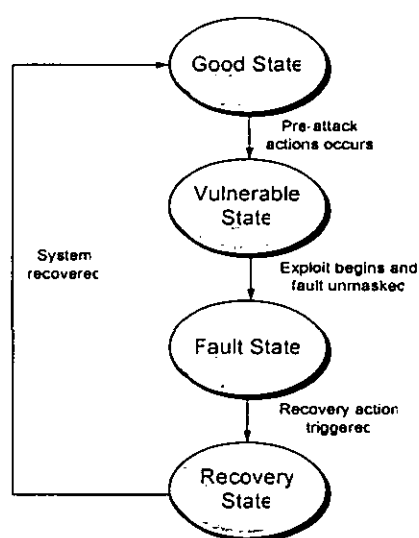


Figure 3-2: State Transition Diagram

### 3.3 Recovery Model

It will often be the case that the effects of a fault will leave the system with greatly reduced resources (processing services, communications capacity, etc) and substantial changes in the services provided to the users will be necessary. The ability to tolerate certain types of fault is the only practical approach to achieve survivability. In this work, there are several fault types: single fault, multiple sequential faults, and multiple concurrent faults for multiple critical services. Besides considering single fault for certain critical service, multiple sequential faults and multiple concurrent faults must be taken into account as well. In multiple sequential faults, the system has to reconfigure the reconfigured system and in multiple concurrent faults, reconfiguration must be simultaneous and consistent in order to maintain stability. In this thesis, reconfiguration refers to resource redistribution and not structural or topological reconfiguration.

The recovery engine defines the requirements for the availability of the system. It guarantees that tasks can be performed exactly at the required moment, access to resources is possible at the required moment and that resources are not demanded unnecessarily or are withheld.

#### 3.3.1 Recovery Action

Figure 3-3 described the recovery action that is mapped into the recovery state. The process starts with diagnosing the destroyed resources of the critical services. The diagnosis part will determine where the damage occurred. The analysis part calculates the amount of damaged resources. In order to move back to good state, the available resources to reconfigure critical services resources must be found and re-allocated.

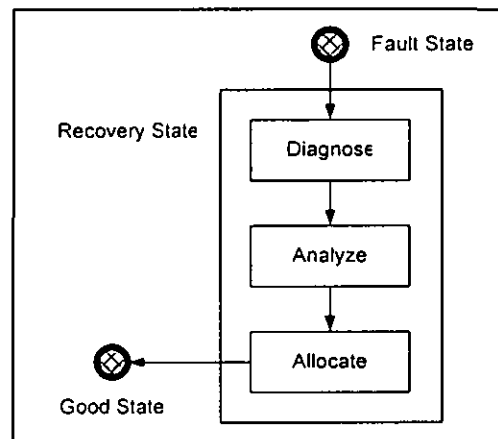


Figure 3-3: Modules in the Recovery Action

There is a priority queue of reconfiguration request inside analysis process. If a multiple concurrent fault occurs, the request will be queued [Mosse, 2003] based on its arrival time and the critical service level. Hence, the analysis will be on a sequential basis as depicted in Figure 3-4.

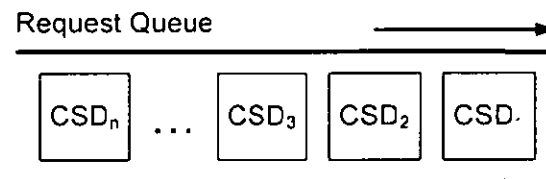


Figure 3-4: Queue Abstract Structures

The process to find the available resource for reconfiguration is based on the tabular method. The four types of resources described in previous sections are defined in three tables, master resources allocation table, critical services table, and non-critical services table. These three tables will be created automatically (dynamic table creation) when the system begins operation, and will be destroyed when the system cease to exist by the OS or

by another process within the current system. For the purpose of our discussion we assume the table creation process is spawned by the recovery engine.

*The master resource allocation table* (Table 3-1) shows the total resources currently used and allocated for redundancy inside the running system. Required resources means the resources that the system needs to run all the services i.e., the total of all resources used by the services, critical and non-critical. Used-redundant means the redundant resources that are currently used by the system's service and unused-redundant means the idle redundant resources that are not currently used by the system.

**Table 3-1 : Master Resource Allocation table**

Resource	$S_1$	$S_2$	$S_a$
Required ( $MQR$ )			
Used Redundant ( $MUR$ )			
Unused Redundant ( $MUNR$ )			

*The critical services resources table* (Table 3-2) shows the resources used by the service, redundant resources that are available for that service, and the resources that are destroyed by fault.

**Table 3-2 : Critical Service Resources table**

Resource	$R_1$	$R_2$	$R_b$
Resource Currently Used ( $CSCU$ )			
Redundant ( $CSR$ )			
Damage ( $CSD$ )			

*The non-critical services resources table* (Table 3-3) shows the resources used by the service and the resources that are released while the services are in progress. Resources are taken-up and released when it is not required.

Table 3-3 : Non-Critical Service Resources table

Resource	$r_1$	$r_2$	$r_c$
Resource Currently Used ( <i>NCSCU</i> )			
Released ( <i>NCSR</i> )			

In the allocation process, there are four scenarios to be analyzed after an error/damage occurs to the critical service resources that find the available resources for reconfiguration. They are:

1. Redundant Resources available with critical service.  
The process will check the redundant resources of the critical service. If it is available, then the problem can be fixed without affecting any other services i.e. the required resources can be used for damage recovery.
2. Redundant resources available with the system.  
If there are insufficient redundant resources with the critical service, it will check the unused redundant resources of the system. If it is available, then allocate the available resources.
3. Released resources available with non-critical services.  
If there are insufficient unused resources of the system, it will check the unused or released resources of the non-critical service. If it is available, then allocate the available resource..
4. Resources are pre-empted from non-critical services.  
If there are insufficient released resources of the non-critical services, the process will check the resources that are being used by the non-critical service to pre-empt them.

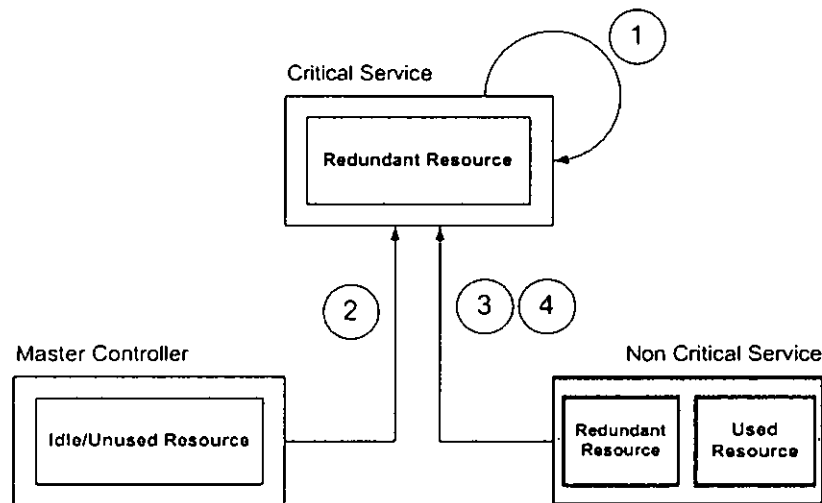


Figure 3-5: Resource Availability and Reallocation

Figure 3-5 depicts all of the possible resource availability for reconfiguring the critical service resource as mentioned earlier.

### 3.3.2 Resource Balancing

The most important factor in resource reconfiguration is resource balancing in order to maintain the stability of the system. Take note that the service is functional when there is at least one active resource to support it [Elnozahy, 2004]. The system has three tables of resources. When the system is activated and the resource allocated, the table will note every allocation into those tables. For example, if there is a critical service, that needs four resources to run the service, then those four resources will be allocated and noted in the critical service's resources table (resource currently used row) and in the master resources allocation table (required resource row). The same goes to non-critical service. Thus, if the engine allocated four resources for critical service and four resources for non-critical service, there will be eight required resources in the master table. For redundant resources of critical service, initially it must be at least equal to one resource.

The tables are updated dynamically at run-time and when fault happens. For example, if there are some resources of the non-critical services that have been released, the used resources count must reflect the change. If some resources of the critical service are destroyed, the used resources count must be deducted as well. The required resource of the master table does not change. In our model it is assumed that there will be no further faults to the affected critical service node while it is being reconfigured.

The following equations describe the situation.

$$CSR_{y,b}(\alpha) \geq 1, NCSCU_{w,c}(\beta) \geq 1 \quad (1)$$

$$MQR_a(\gamma) = \sum_{y=1}^n CS_{y,b}(\alpha) + \sum_{w=1}^n NCS_{w,c}(\beta), \text{ where } a = b = c \quad (2)$$

$$CS_{y,b}(\alpha) = CSCU_{y,b}(\alpha) + (CSR_{y,b}(\alpha) - CSCU_{y,b}(\alpha)) + CSD_{y,b}(\alpha) \quad (3)$$

$$NCS_{w,c}(\beta) = NCSCU_{w,c}(\beta) + (NCSR_{w,c}(\beta) - NCSCU_{w,c}(\beta)) \quad (4)$$

The redundant resources of critical services and currently used resources of non-critical services must be at least equal to 1 unit are shown in (1). The required resources of master table are the total of required resources of critical services and redundant resources of non-critical services, as shown in (2). Equations (3) and (4) show the required resources, both used and redundant of the critical and non-critical services.

### 3.3.3 Self Reconfiguration

Koopman (2003) defined four general aspect categories of self-healing system problem space: fault model, system response, system completeness, and design context [Koopman, 2003]. Recovery is placed under system response category. The proposed recovery model in this thesis is developed to recover a faulted service by reconfiguring the damaged resources.



In other word, by combining all those ideas, the model can be called as self-reconfiguration model for survivable system.

### 3.3.4 Diagnosis and Analysis Process

In this section, the process of recovery will be explained in detail, from diagnosis to the allocation process. After diagnosing the error and identifying where the error has taken place, the analysis process will queue the request in a reconfiguration request priority queue. If  $CSD_{y,b}$  is the damaged critical service resources  $y$  on resource  $b$ ,  $CSD_{y,b}(\alpha)$  shows the damaged amount,  $q_i$  is the arrival time of  $i$ th damage to the reconfiguration request queue,  $l_i$  is the level of  $i$ th damaged critical service and  $Q$  is the queue, the queue will be:

$$Q : \{(CSD_{y,b}(\alpha), q_0, l_0), (CSD_{y,b}(\alpha), q_1, l_1), \dots, (CSD_{y,b}(\alpha), q_i, l_i)\}$$

When a multiple concurrent fault occurs, the resource allocation will be done in a concurrent manner, as shown in Figure 3-6. Once the sequential analysis for one fault is completed the process moves on to the allocation process. The search for and re-allocation will be processed simultaneously for all the requests in the queue.

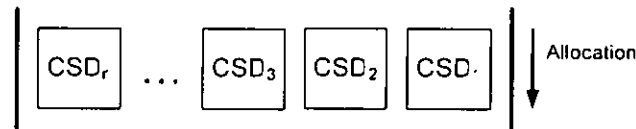


Figure 3-6: Concurrent Allocation Queue

## 3.3.5 Allocation Process

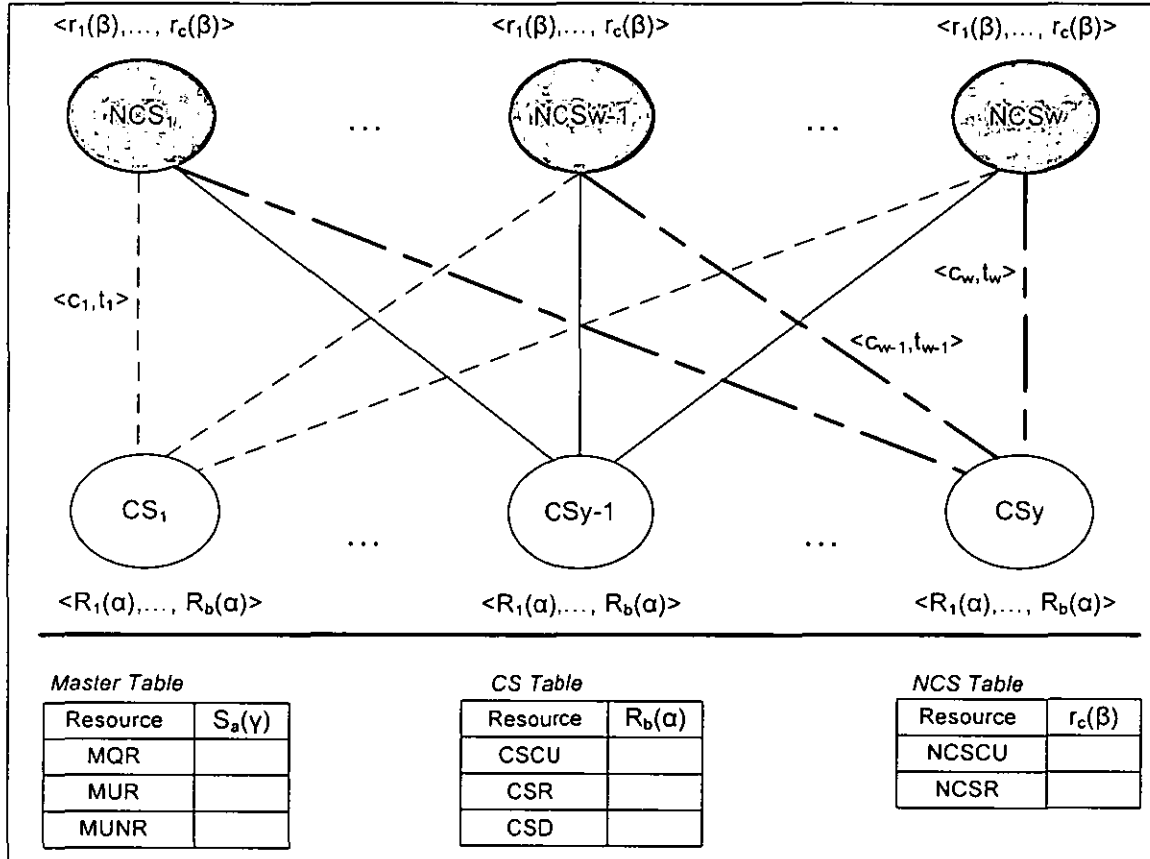


Figure 3-7: Graph of The Services

In Figure 3-7, the Critical Service connected to Non-critical Service by cost of taking the *NCS* resource and *NCS* response time in a tuple  $\langle c, t \rangle$ . The amount of master resource notated in  $\gamma$ , *CS* resource notated in  $\alpha$  and *NCS* resource notated in  $\beta$ .

The target is to find the feasible scheme of reconfiguration ( $\delta$ ), where  $\Omega = \{\delta_1, \dots, \delta_i\}$ , from possible available resource with minimum response time ( $T$ ), minimum cost ( $C$ ) and

minimum reconfiguration resource number( $N$ ). Let  $\Omega$  be the set of all possible feasible scheme. Hence the basis model will be:

$$\min N(\delta), \delta \in \Omega$$

$$\min T(\delta), \delta \in \Omega$$

$$\min C(\delta), \delta \in \Omega$$

To choose the reconfiguration scheme based on the basis model, in hierarchical order, number of resource comes in as the first priority, then the response time and last the cost of reconfiguration. This research does not focus on choosing an optimal decision.

There are some scenarios and sub-scenarios in finding the available resource based on Table 3-1, Table 3-2, and Table 3-3:

**Scenario 1:** Redundant Resources available with critical service.

One of the resources  $R_i$  of a critical service is destroyed. It requires some resources to fix it. If there are enough redundant resources from the critical service, the problem can be easily fixed without affecting any other services.

$CSD_{y,b}(\alpha)$  has been destroyed, where  $\alpha \geq 1$ . If there is  $CSR_{y,b}(\alpha)$  such that  $CSR_{y,b}(\alpha) \geq CSD_{y,b}(\alpha)$  then allocate  $CSR_{y,b}(\alpha)$ . In this scenario, the response time and cost factor are not explicitly considered. Thus:

$$N(\delta) = 1$$

$$T(\delta) = t_{CSR_{y,b}}$$

$$C(\delta) = c_{CSR_{y,b}}$$

$$\delta = CSR_{y,b}$$

Example:

Supposed that  $CSD_{1,2}(\alpha) = 5$ ,  $t_{CSR_{1,2}} = 3$  units time,  $c_{CSR_{1,2}} = 7$  units cost and  $CSR_{1,2}(\alpha) = 6$ . Since  $CSR_{y,b}(\alpha) \geq CSD_{y,b}(\alpha)$ , then allocate 5 from  $CSR_{1,2}(\alpha)$ . Therefore, the reconfiguration scheme is  $\delta = CSR_{1,2}$ , the reconfiguration time is  $T(\delta) = 3$  units time, the reconfiguration cost is  $C(\delta) = 35$  units cost, and the number of involved resource is  $N(\delta) = 1$ .

**Scenario 2:** Redundant resources available with the system.

One of the critical resources  $R$  is destroyed. It requires some resources to fix it. Unfortunately there are only small number redundant resources. In this case, the recovery process will be fixed by adding resources from the unused/idle redundant resources of the system; from the master resource allocation controller.

$CSD_{y,b}(\alpha)$  has been destroyed, where  $\alpha \geq 1$ . If there is  $MUNR_a(\gamma)$  such that  $MUNR_a(\gamma) \geq CSD_{y,b}(\alpha)$  then allocate  $MUNR_a(\gamma)$ . Similar as scenario 1, here the response time and cost factor are not explicitly considered.

$$N(\delta) = 1$$

$$T(\delta) = t_{MUNR_a}$$

$$C(\delta) = c_{MUNR_a}$$

$$\delta = MUNR_a$$

Example:

Supposed that  $CSD_{1,2}(\alpha) = 5$ , no available  $CSD_{1,2}(\alpha)$ ,  $MUNR_2(\gamma) = 6$ ,  $t_{MUNR_2} = 3$  units time,  $c_{MUNR_2} = 5$  units cost. Since  $MUNR_a(\gamma) \geq CSD_{y,b}(\alpha)$ , then allocate 5 from  $MUNR_a(\gamma)$ . Therefore the reconfiguration scheme is  $\delta = MUNR_2$ , the reconfiguration time is  $T(\delta) = 3$  units time, the reconfiguration cost is  $C(\delta) = 25$  units cost, and the number of involved resource is  $N(\delta) = 1$ .

**Scenario 3:** Released resources available with non-critical services.

One of critical resources  $R$  is destroyed. It requires some resources to fix it. If there is not enough redundant resources of critical service and unused redundant of the system, the problem will be fixed if the recovery process takes another resource from released resource of non-critical service resources.

$CSD_{y,b}(\alpha)$  has been destroyed, where  $\alpha \geq 1$ . If scenario 1 and 2 fail, then look into the released resource of  $NCS$ . There are three possibilities in this scenario:

**Scenario 3.1:** Released resources available with only one non-critical service.

$CSD_{y,b}(\alpha)$  has been destroyed, where  $\alpha \geq 1$ . If there is only one  $NCSR_{w,c}(\beta)$  such that  $NCSR_{w,c}(\beta) \geq CSD_{y,b}(\alpha)$  then allocate  $NCSR_{w,c}(\beta)$ . Thus:

$$N(\delta) = 1$$

$$T(\delta) = t_{NCSR_{w,c}}$$

$$C(\delta) = c_{NCSR_{w,c}}$$

$$\delta = NCSR_{w,c}$$

Example:

Supposed that  $CSD_{1,2}(\alpha) = 5$ , no available  $CSD_{1,2}(\alpha)$ , no available  $MUNR_2(\gamma)$ ,  $NCSR_{1,2}(\beta) = 6$ ,  $t_{NCSR_{1,2}} = 3$  units time, and  $c_{NCSR_{1,2}} = 5$  units cost. Since  $NCSR_{w,c}(\beta) \geq CSD_{y,b}(\alpha)$ , then allocate 5 from  $NCSR_{1,2}(\beta)$ . Therefore the reconfiguration scheme is  $\delta = NCSR_{1,2}$ , the reconfiguration time is  $T(\delta) = 3$  units time, the reconfiguration cost is  $C(\delta) = 25$  units cost, and the number of involved resource is  $N(\delta) = 1$ .

**Scenario 3.2 :** Released resources available with one non-critical service.

$CSD_{y,b}(\alpha)$  has been destroyed, where  $\alpha \geq 1$ . If there are some  $NCSR_{w,c}(\beta)$  such that each  $NCSR_{w,c}(\beta) \geq CSD_{y,b}(\alpha)$  then allocate  $NCSR_{w,c}(\beta)$  with condition resource response time  $\min(t_1, \dots, t_w)$  and resource cost is  $\min(c_1, \dots, c_w)$ . Thus:

$$N(\delta) = 1$$

$$T(\delta) = \min(t_1, \dots, t_w)$$

$$C(\delta) = \min(c_1, \dots, c_w)$$

$$\delta = NCSR_{w,c}$$

Example:

Supposed that  $CSD_{1,2}(\alpha) = 5$ , no available  $CSD_{1,2}(\alpha)$ , no available  $MUNR_2(\gamma)$ , and there are three possibilities  $NCSR_{w,c}$ :

$$NCSR_{1,2}(\beta) = 6, t_{NCSR_{1,2}} = 3 \text{ units time, } c_{NCSR_{1,2}} = 5 \text{ units cost}$$

$$NCSR_{2,2}(\beta) = 7, t_{NCSR_{2,2}} = 6 \text{ units time, } c_{NCSR_{2,2}} = 5 \text{ units cost}$$

$$NCSR_{3,2}(\beta) = 8, t_{NCSR_{3,2}} = 2 \text{ units time, } c_{NCSR_{3,2}} = 6 \text{ units cost}$$

All of  $NCSR_{w,c}(\beta) \geq CSD_{y,b}(\alpha)$ , in this condition, 5 units resource are allocated from  $NCSR_{3,2}(\beta)$  because it has the minimum response time. Therefore the reconfiguration scheme is  $\delta = NCSR_{3,2}$ , the reconfiguration time is  $T(\delta) = 2$  units time, the reconfiguration cost is  $C(\delta) = 30$  units cost, and the number of involved resource is  $N(\delta) = 1$ .

**Scenario 3.3:** Released resources available with combination of non-critical services.

$CSD_{y,b}(\alpha)$  has been destroyed, where  $\alpha \geq 1$ . If there are some  $NCSR_{w,c}(\beta)$  such that each  $NCSR_{w,c}(\beta) < CSD_{y,b}(\alpha)$ , the allocation process can combine those  $NCSR_{w,c}(\beta)$  then satisfied  $\sum_{w=1}^n NCSR_{w,c}(\beta) \geq CSD_{y,b}(\alpha)$ . So there will be some  $\delta_1, \dots, \delta_i$  with  $C(\delta_i) = c_1 + \dots + c_w$  as the cost of each combination and  $T(\delta_i) = \max(t_1, \dots, t_w)$  as the response time of the combination, then allocate  $\sum_{w=1}^n NCSR_{w,c}(\beta)$  with condition  $\min(C(\delta_1), \dots, C(\delta_i))$ ,  $\min(T(\delta_1), \dots, T(\delta_i))$  and  $\min(N(\delta_1), \dots, N(\delta_i))$ . Thus:

$$N(\delta) = \min(N(\delta_1), \dots, N(\delta_i))$$

$$T(\delta) = \min(T(\delta_1), \dots, T(\delta_i))$$

$$C(\delta) = \min(C(\delta_1), \dots, C(\delta_i))$$

$$\delta = \{NCSR_{1,1}, \dots, NCSR_{w,c}\}$$

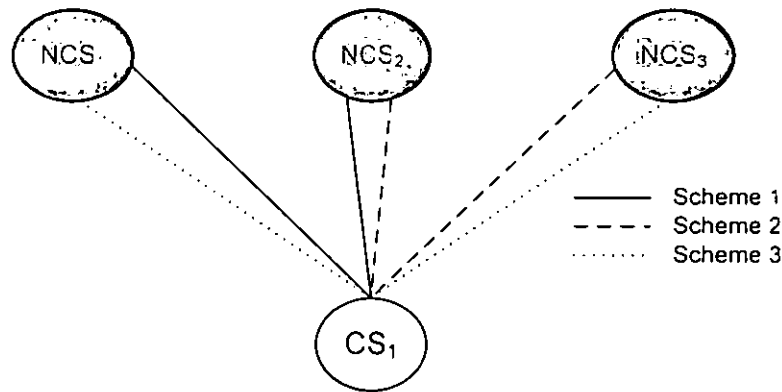


Figure 3-8: Possible schemes for Scenario 3.3



Example:

Supposed that  $CSD_{1,2}(\alpha) = 5$ , no available  $CSD_{1,2}(\alpha)$ , no available  $MUNR_2(\gamma)$ , and there are three  $NCSR_{w,c}$  that possible to be combined:

$$NCSR_{1,2}(\beta) = 2, t_{NCSR_{1,2}} = 3 \text{ units time}, c_{NCSR_{1,2}} = 5 \text{ units cost}$$

$$NCSR_{2,2}(\beta) = 3, t_{NCSR_{2,2}} = 6 \text{ units time}, c_{NCSR_{2,2}} = 5 \text{ units cost}$$

$$NCSR_{3,2}(\beta) = 4, t_{NCSR_{3,2}} = 4 \text{ units time}, c_{NCSR_{3,2}} = 6 \text{ units cost}$$

$$\text{Scheme 1: } \delta_1 = \{(NCSR_{1,2}), (NCSR_{2,2})\},$$

$$\text{where } NCSR_{1,2}(\beta) = 2 \text{ and } NCSR_{2,2}(\beta) = 3. T(\delta_1) = 6, C(\delta_1) = 10 + 15 = 25.$$

$$\text{Scheme 2: } \delta_2 = \{(NCSR_{1,2}), (NCSR_{3,2})\},$$

$$\text{where } NCSR_{1,2}(\beta) = 2 \text{ and } NCSR_{3,2}(\beta) = 3. T(\delta_2) = 4, C(\delta_2) = 10 + 18 = 28.$$

$$\text{Scheme 3: } \delta_3 = \{(NCSR_{2,2}), (NCSR_{3,2})\},$$

$$\text{where } NCSR_{2,2}(\beta) = 3 \text{ and } NCSR_{3,2}(\beta) = 2. T(\delta_3) = 6, C(\delta_3) = 15 + 8 = 23.$$

In this condition, since all scheme has the same number of involved resource, 5 units resource are allocated from  $\delta_2$  because it has the minimum response time. Therefore the reconfiguration scheme is  $\delta = \{(NCSR_{1,2}), (NCSR_{3,2})\}$ , the reconfiguration time is  $T(\delta) = 4$  units time, the reconfiguration cost is  $C(\delta) = 28$  units cost, and the number of involved resource is  $N(\delta) = 2$ .

**Scenario 4:** Resources are pre-empted from non-critical services.

One of critical resources  $R$  is destroyed. It requires some resources to fix it. If there is not enough redundant resources of critical service, unused redundant of the system and released resource of non-critical service, the problem will be fixed if the recovery process pre-empts some resources from non-critical service resources that currently used.

$CSD_{y,b}(\alpha)$  has been destroyed, where  $\alpha \geq 1$ . If none of the scenario 1, 2, and 3 can fulfil the recovery, then pre-empt from the currently used resource of  $NCS$ . There are three possibilities in this scenario:

**Scenario 4.1:** Resources are pre-empted from only one non-critical service.

$CSD_{y,b}(\alpha)$  has been destroyed, where  $\alpha \geq 1$ . If there is only one  $NCSCU_{w,c}(\beta)$  such that  $NCSCU_{w,c}(\beta) \geq CSD_{y,b}(\alpha)$  then allocate  $NCSCU_{w,c}(\beta)$ . Thus:

$$N(\delta) = 1$$

$$T(\delta) = t_{NCSCU_{w,c}}$$

$$C(\delta) = c_{NCSCU_{w,c}}$$

$$\delta = NCSCU_{w,c}$$

Example:

Supposed that  $CSD_{1,2}(\alpha) = 5$ , no available  $CSD_{1,2}(\alpha)$ , no available  $MUNR_2(\gamma)$ , no available  $NCSR_{y,2}(\beta)$ , and  $NCSCU_{1,2}(\beta) = 6$ ,  $t_{NCSCU_{1,2}} = 3$  units time, and  $c_{NCSCU_{1,2}} = 5$  units cost. Since  $NCSCU_{w,c}(\beta) \geq CSD_{y,b}(\alpha)$ , then allocate 5 from  $NCSR_{1,2}(\beta)$ . Therefore the reconfiguration scheme is  $\delta = NCSCU_{1,2}$ , the reconfiguration time is  $T(\delta) = 3$  units time, the

reconfiguration cost is  $C(\delta) = 25$  units cost, and the number of involved resource is  $N(\delta) = 1$ .

**Scenario 4.2:** Resources are pre-empted from one non-critical service.

$CSD_{y,b}(\alpha)$  has been destroyed, where  $\alpha \geq 1$ . If there are some  $NCSCU_{w,c}(\beta)$  such that each  $NCSCU_{w,c}(\beta) \geq CSD_{y,b}(\alpha)$  then allocate  $NCSCU_{w,c}(\beta)$  with resource response time  $\min(t_1, \dots, t_w)$  and resource cost is  $\min(c_1, \dots, c_w)$ . Thus:

$$N(\delta) = 1$$

$$T(\delta) = \min(t_1, \dots, t_w)$$

$$C(\delta) = \min(c_1, \dots, c_w)$$

$$\delta = NCSCU_{w,c}$$

Example:

Supposed that  $CSD_{1,2}(\alpha) = 5$ , no available  $CSD_{1,2}(\alpha)$ , no available  $MUNR_2(\gamma)$ , no available  $NCSCR_{y,2}(\beta)$ , and there are three possibilities  $NCSCU_{w,c}$ :

$$NCSCU_{1,2}(\beta) = 6, t_{NCSCU_{1,2}} = 3 \text{ units time}, c_{NCSCU_{1,2}} = 5 \text{ units cost}$$

$$NCSCU_{2,2}(\beta) = 7, t_{NCSCU_{2,2}} = 6 \text{ units time}, c_{NCSCU_{2,2}} = 5 \text{ units cost}$$

$$NCSCU_{3,2}(\beta) = 8, t_{NCSCU_{3,2}} = 2 \text{ units time}, c_{NCSCU_{3,2}} = 6 \text{ units cost}$$

All of  $NCSCU_{w,c}(\beta) \geq CSD_{y,b}(\alpha)$ , in this condition, 5 units resource are allocated from  $NCSCU_{3,2}(\beta)$  because it has the minimum response time. Therefore the reconfiguration scheme is  $\delta = NCSCU_{3,2}$ , the reconfiguration time is  $T(\delta) = 2$  units time, the reconfiguration cost is  $C(\delta) = 30$  units cost, and the number of involved resource is  $N(\delta) = 1$ .

**Scenario 4.3:** Resources are pre-empted from combination of non-critical services.

$CSD_{y,b}(\alpha)$  has been destroyed, where  $\alpha \geq 1$ . If there are some  $NCSCU_{w,c}(\beta)$  such that each  $NCSCU_{w,c}(\beta) < CSD_{y,b}(\alpha)$ , we can combine those  $NCSCU_{w,c}(\beta)$  then satisfied

$\sum_{w=1}^n NCSCU_{w,c}(\beta) \geq CSD_{y,b}(\alpha)$ . So there will be some  $\delta_1, \dots, \delta_i$  with  $C(\delta_i) = c_1 + \dots + c_w$  as the cost of each combination and  $T(\delta_i) = \max(t_1, \dots, t_w)$  as the response time of the combination,

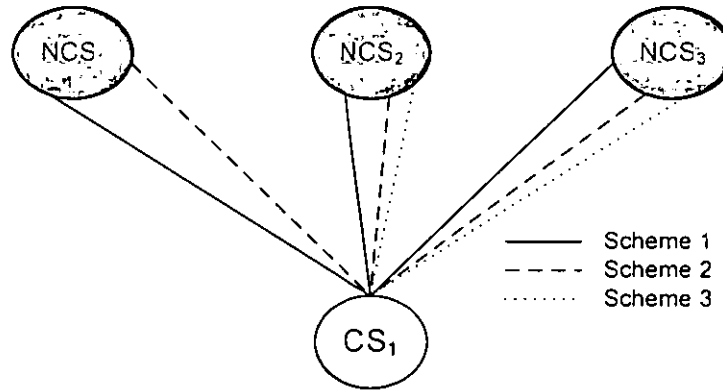
then allocate  $\sum_{w=1}^n NCSCU_{w,c}(\beta)$  with condition  $\min(C(\delta_1), \dots, C(\delta_i))$  and  $\min(T(\delta_1), \dots, T(\delta_i))$ . Thus:

$$N(\delta) = \min(N(\delta_1), \dots, N(\delta_i))$$

$$T(\delta) = \min(T(\delta_1), \dots, T(\delta_i))$$

$$C(\delta) = \min(C(\delta_1), \dots, C(\delta_i))$$

$$\delta = \{NCSCU_{1,1}, \dots, NCSCU_{w,c}\}$$



**Figure 3-9:** Possible schemes for Scenario 4.3

Example:

It is supposed that  $CSD_{1,2}(\alpha) = 5$ , no available  $CSD_{1,2}(\alpha)$ , no available  $MUNR_2(\gamma)$ , no available  $NCSCR_{y,2}(\beta)$ , and there are three  $NCSCU_{w,c}$  that possible to be combined:

$$NCSCU_{1,2}(\beta) = 2, t_{NCSCU_{1,2}} = 3 \text{ units time}, c_{NCSCU_{1,2}} = 5 \text{ units cost}$$

$$NCSCU_{2,2}(\beta) = 3, t_{NCSCU_{2,2}} = 6 \text{ units time}, c_{NCSCU_{2,2}} = 5 \text{ units cost}$$

$$NCSCU_{3,2}(\beta) = 4, t_{NCSCU_{3,2}} = 4 \text{ units time}, c_{NCSCU_{3,2}} = 6 \text{ units cost}$$

$$\text{Scheme 1: } \delta_1 = \{(NCSCU_{1,2}), (NCSCU_{2,2}), (NCSCU_{3,2})\},$$

$$\text{where } NCSCU_{1,2}(\beta) = 1, NCSCU_{2,2}(\beta) = 2 \text{ and } NCSCU_{3,2}(\beta) = 2. T(\delta_1) = 6, C(\delta_1) = 5 + 10 + 12 = 27.$$

$$\text{Scheme 2: } \delta_2 = \{(NCSCU_{1,2}), (NCSCU_{3,2}), (NCSCU_{2,2})\},$$

$$\text{where } NCSCU_{1,2}(\beta) = 1, NCSCU_{3,2}(\beta) = 3 \text{ and } NCSCU_{2,2}(\beta) = 1. T(\delta_2) = 6, C(\delta_2) = 5 + 18 + 5 = 28.$$

$$\text{Scheme 3: } \delta_3 = \{(NCSCU_{2,2}), (NCSCU_{3,2})\},$$

$$\text{where } NCSCU_{2,2}(\beta) = 2 \text{ and } NCSCU_{3,2}(\beta) = 3. T(\delta_3) = 6, C(\delta_3) = 10 + 18 = 28.$$

In this condition, 5 units resource are allocated from  $\delta_3$  because it has the minimum number of involved resources. Therefore the reconfiguration scheme is  $\delta = \{(NCSCU_{2,2}), (NCSCU_{3,2})\}$ , the reconfiguration time is  $T(\delta) = 6$  units time, the reconfiguration cost is  $C(\delta) = 28$  units cost, and the number of involved resources is  $N(\delta) = 2$ .

### 3.3.6 General Interconnected Network Recovery Model

It will often be a case that a system will run out of resources when multiple concurrent faults occur. Running out of resources means all the possible resources are already “booked” for other reconfigurations or are currently being used. In this circumstance, the faulted system will broadcast the resources needed for reconfiguration in the network. The broadcasted parameter will be captured and compared with only the Master Unused Resource (*MUNR*). In the recovery process, the other system request will be queued as normal. The available *MUNR* amount will be retrieved in allocation process.

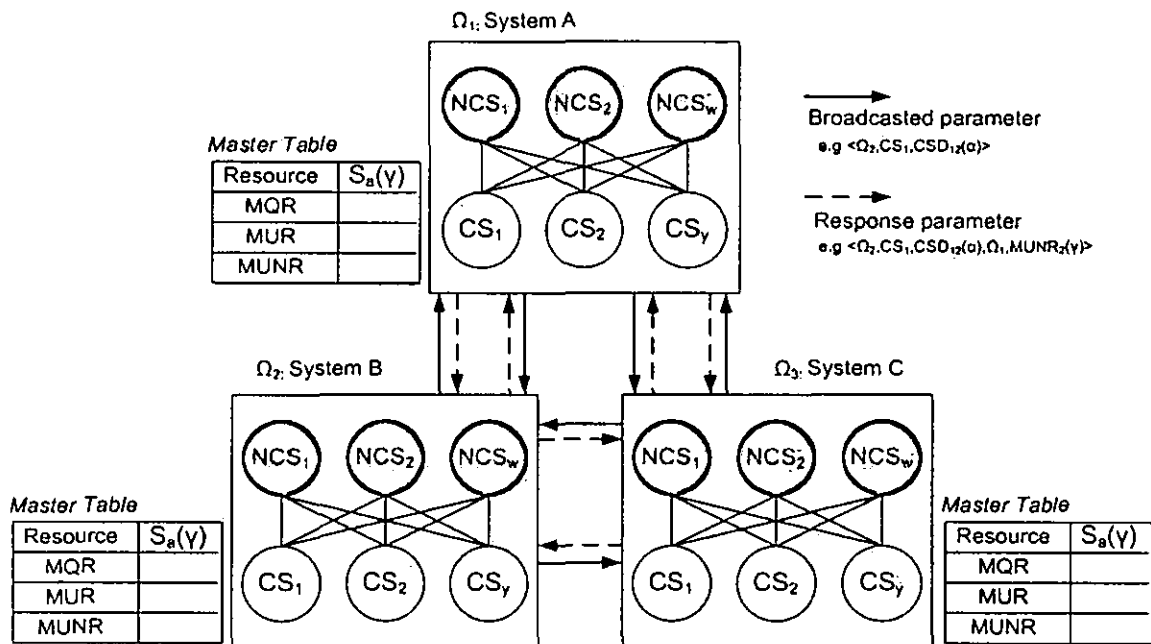


Figure 3-10: the Interconnected Network

Figure 3-10 shows the abstract interconnected networks graph. It has three systems with the same services structure. Each system has a recovery engine to perform the resource reconfiguration. The system consist of a set of network is  $\Omega = \{\Omega_1, \Omega_2, \dots, \Omega_i\}$ . The

*broadcasted parameter* will contain  $\langle \Omega_i, CS_y, CSD_{y,b}(\alpha) \rangle$ , where  $\Omega_i$  is the faulted system that sends the parameter,  $CS_y$  is the destroyed service, and  $CSD_{y,b}(\alpha)$  is the damaged service with its amount. The *response parameter* will contain  $\langle \Omega_i, CS_y, CSD_{y,b}(\alpha), \Omega_n, MUNR_n(\gamma) \rangle$ , where  $\Omega_i$  is the system that responds the parameter, and  $MUNR_n(\gamma)$  is the resource amount of  $\Omega_n$  that will be used for the reconfiguration. To resolve this situation, initially, it is assumed that there is an at least one system respond the message with available resources. This assumption is raised because the system has no fail state. There are three other cases:

**Case 1:** Only one system responds with sufficient resources.

In this case, the faulted system will obviously use the  $MUNR_b(\gamma)$  of the system that responds for reconfiguring the damaged resource.

Example:

There is a broadcasted parameter  $\langle \Omega_1, CS_1, CSD_{1,2}(4) \rangle$  and only one system responds by  $\langle \Omega_1, CS_1, CSD_{1,2}(4), \Omega_2, MUNR_2(4) \rangle$  where  $t_{MUNR_2^{n_2}} = 3$  units time and  $c_{MUNR_2^{n_2}}$  of  $\Omega_2 = 3$  units cost. In this case, the  $MUNR_2(4)$  of  $\Omega_2$  will be re-allocated. Therefore, the reconfiguration scheme is  $\delta = MUNR_2^{n_2}$ , the reconfiguration time is  $T(\delta) = 3$  units time, the reconfiguration cost is  $C(\delta) = 3$  units cost, and the number of involved resources is  $N(\delta) = 1$ .

Case 2: There are more than one systems respond each with at least the required number of resources.

In this case, the faulted system will then select the resource that satisfies the requisite condition; response time and cost.

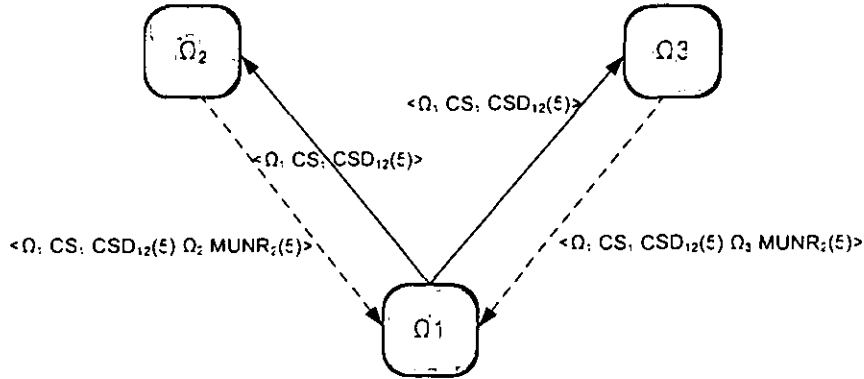


Figure 3-11: Interconnected Network Recovery for Case 2

Example:

System A ( $\Omega_1$ ) broadcasts a request parameter  $\langle \Omega_1, CS_1, CSD_{1,2}(5) \rangle$  within the network as in Figure 3-11. The parameter indicates 5 units of resource 2 from critical service 1 of system A. There are two systems ( $\Omega_2$  and  $\Omega_3$ ) that have the needed resource. If  $t_{MUNR_1^2} = 3$  unit time,  $c_{MUNR_2^2} = 6$  unit cost,  $t_{MUNR_3^2} = 4$  unit time, and  $c_{MUNR_3^2} = 4$  unit cost, the recovery process will take the  $MUNR_2^2$ , master controller resource from  $\Omega_2$  because it has the minimum response time.



**Case 3:** There are more than one systems respond each with less than the required number of resources. In this case, a scheme has to be devised to select the best combination.

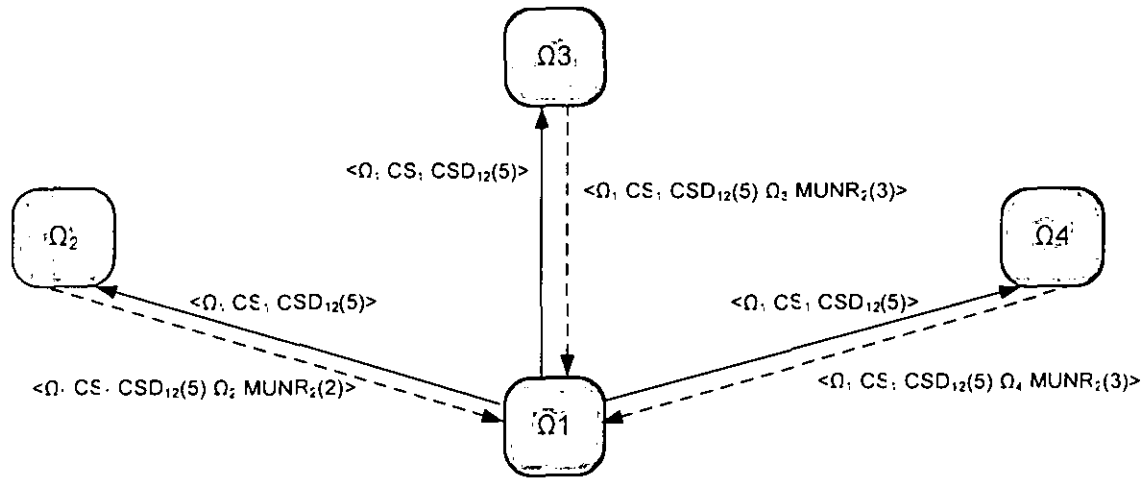


Figure 3-12 : Interconnected Network Recovery for Case 3

Example:

System A ( $\Omega_1$ ) broadcasts a request parameter  $\langle \Omega_1, CS_1, CSD_{1,2}(5) \rangle$  within the network as in Figure 3-12 : Interconnected Network Recovery for Case 3. The parameter indicates 5 units of resource 2 from critical service 1 of system A. There are three systems ( $\Omega_2$ ,  $\Omega_3$  and  $\Omega_4$ ) that respond the message. If  $t_{MUNR_2^2} = 3$  units time,  $c_{MUNR_2^2} = 7$  units cost,  $t_{MUNR_2^3} = 4$  units time,  $c_{MUNR_2^3} = 6$  units cost,  $t_{MUNR_2^4} = 5$  units time, and  $c_{MUNR_2^4} = 6$  units cost, the recovery will have four combinations of  $MUNR$ :

*Scheme 1:*  $\delta_1 = \{MUNR_2^2, MUNR_2^3\}$ ,

where  $MUNR_2^2(\gamma) = 2$  and  $MUNR_2^3(\gamma) = 3$ ,  $T(\delta_1) = 4$ ,  $N(\delta_1) = 14 + 18 = 32$ .

*Scheme 2:*  $\delta_2 = \{MUNR_2^2, MUNR_2^4\}$ ,

where  $MUNR_2^2(\gamma) = 2$  and  $MUNR_2^4(\gamma) = 3$ ,  $T(\delta_2) = 5$ ,  $N(\delta_2) = 14 + 18 = 32$ .

*Scheme 3:*  $\delta_3 = \{MUNR_2^3, MUNR_2^4\}$ ,

where  $MUNR_2^2(\gamma) = 3$  and  $MUNR_2^2(\gamma) = 2$ ,  $T(\delta_3) = 5$ ,  $N(\delta_3) = 18 + 12 = 30$ .

*Scheme 4:*  $\delta_4 = \{MUNR_2^3, MUNR_2^4\}$ ,

where  $MUNR_2^2(\gamma) = 2$  and  $MUNR_2^2(\gamma) = 3$ ,  $T(\delta_3) = 5$ ,  $N(\delta_3) = 12 + 18 = 30$ .

In this scenario, 5 units resource are allocated from  $\delta_1$  to the faulted system  $\Omega_1$  because it has the minimum response time.

### 3.3.7 Recovery Algorithm

In this section, the full algorithm is provided in pseudo-code form. It starts with initial condition where there is no damage, scenarios process, and then diagnoses, analysis, and allocation respectively. See Figure 3-13.

```
1. Initially set CSD = null
2. Let CSD be the input of the process
3. Diagnose(CSD); Set Queue; Analyze(CSD)
4. while (not CSD = 0) do
  4.1 if CS_redundant > CSD then
    4.1.1 allocate(CS_redundant)
    4.1.2 else allocate(CS_redundant);
    4.1.3 CSD := CSD - CS_redundant;
    4.1.4 time1 := CS_redundant_time;
    4.1.5 cost1 := CS_redundant_cost;
  4.2 if system_unused_redundant > CSD then
    4.2.1 allocate(system_unused_redundant)
    4.2.2 else allocate(system_unused_redundant);
    4.2.3 CSD := CSD - system_unused_redundant;
    4.2.4 time2 := system_unused_redundant_time;
    4.2.5 cost2 := system_unused_redundant_cost;
  4.3 if NCS_redundant > CSD then
    4.3.1 allocate(NCS_redundant)
    4.3.2 else allocate(NCS_redundant);
    4.3.3 CSD := CSD - NCS_redundant;
    4.3.4 time3 := NCS_redundant_time;
    4.3.5 cost3 := NCS_redundant_cost;
  4.4 if NCS_used > CSD then
    4.4.1 allocate(NCS_used)
    4.4.2 else allocate(NCS_used);
    4.4.3 CSD := CSD - NCS_used;
    4.4.4 time4 := NCS_used_time;
    4.4.5 cost4 := NCS_used_cost;
  4.5 broadcast(CSD)
    4.5.1 allocate(other_system_unused_redundant);
    4.5.2 CSD := CSD - other_system_unused_redundant;
    4.5.3 time5 := other_system_unused_redundant_time;
    4.5.4 cost5 := other_system_unused_redundant_cost;
5. num := count(resource_used);
6. time := max(time1,time2,time3,time4,time5);
7. cost := sum(cost1,cost2,cost3,cost4,cost5);
8. output := possible_scheme(min(num),min(time),min(cost));
9. final_time := time_of_scheme;
10. final_cost := cost_of_scheme;
```

Figure 3-13: Recovery Algorithm

The flowchart of the algorithm is placed in APPENDIX. Since optimization is beyond the scope of this research, the engine does not incorporate optimization functions based on response time, resource cost, and resource amount.

### 3.4 Summary

This thesis is proposing a different model of recovery in the fault tolerance area. The model deals with destroyed critical service in a system and considering some scenarios to reconfigure the resources. The consideration of all the possible available resources and requisite conditions are taken to build a survivable system.

## CHAPTER 4 : SPECIFICATION AND SIMULATION MODEL

The developed model needs to be proven. A widely used formal specification language was chosen as the engine state proofing. The development a simulation model was created to affirm the model. A further discussion on some case studies is carried out.

### 4.1 Z Notation

The Z notation is one of the best known formal methods which is often declared as a formal specification language and which is gaining widespread acceptance as a useful means of specifying software systems. The Z notation is a descriptive method with a medium level of formality. That is the method can be used together with computer-aided tools. In the sense of a specification language it is model oriented, based upon the set theory and mathematical logic, i.e. it is based on model design which is given by description of the system state and operations over this state. The representation, structure and meaning of the formal part of specification, written in the Z notation, is defined in the draft standard.

In the Z notation there are two languages: the mathematical language and the schema language. The former is used to describe different aspects of design: objects and the relationships between them. The latter is used to structure and compose descriptions: collating pieces of information, encapsulating them and naming them for re-use [Woodcock, 1996].

## 4.2 Formal Z Notation

In this section a way of Z specification design is presented with brief explanation of particular parts. First is the declaration of the basic set types of the processes. The model need to deal with damaged resource and redundant resource. For reconfiguration purposes, it will not matter what form these damages and resources take, hence the set of damage and redundant will be:

$[DAMAGE, RESOURCE]$

Without saying what kind of object they contain is allowed in Z to be named to those sets. The first aspect of the system to describe is its state space. It is shown in a schema:

<i>RecoveryEngine</i>
<i>csd</i> : $\mathbb{P} \text{ DAMAGE}$ <i>recover</i> : $DAMAGE \rightarrow RESOURCE$
<i>csd</i> = dom <i>recover</i>

This consists of a part above the central dividing line, in which some variables are declared, and a part below the line which gives a relationship between the values of the variables. *csd* is the set of damages and *recover* is a function which, when applied to certain damages, gives the redundant resource to repair it.

<i>Diagnose</i>
$\exists \text{RecoveryEngine}$ $\text{csd!} : \mathbf{P} \text{ DAMAGE}$
<hr/> $\text{csd!} = \text{csd}$

The first operation in the recovery engine is diagnosing where the damage has taken placed. For that operation, the engine makes no state changes, indicated by  $\exists$  notation, and only checks from the set of *DAMAGE* to get *csd!* as the output.

<i>Δqueue</i>
$\text{items} : \text{seq } \text{DAMAGE}$ $\text{items}' : \text{seq } \text{DAMAGE}$

<i>Analyze</i>
$\exists \text{RecoveryEngine}$ $\text{csd!} : \mathbf{P} \text{ DAMAGE}$ $\Delta \text{queue}$ $\text{item?}, \text{item!} : \text{DAMAGE}$
<hr/> $\exists \text{items}'' : \text{seq } \text{DAMAGE}$ <ul style="list-style-type: none"> <li><math>\text{items}'' = \text{items} \hat{\ } \langle \text{item?} \rangle \wedge \text{items}'' \neq \diamond \wedge \text{items}'' = \langle \text{item!} \rangle \hat{\ } \text{items}'</math></li> </ul> $\text{csd!} = \{\text{item!}\}$

In general process, the analysis part is akin with diagnosis part. There is no state change within the process. The schema has a queue process that has state change, indicated by  $\Delta$ , before queue and after queue, consists of add items and leave items. The output below the line is the amount of *csd!*, result of the queue.

<i>Allocate</i>
$\Delta RecoveryEngine$ $dam?: DAMAGE$ $reds?: RESOURCE$
$dam? \in csd$ $recover' = recover \oplus \{(dam? \mapsto reds?)\}$

Next is the allocation part. Here, the schema formed in general condition of each scenario. Since this process has a significant result, move to good state, hence it has a state change of the recovery engine.

<i>initRecoveryEngine</i>
$RecoveryEngine$
$csd = \emptyset$

To finish the specification, the initial state of the recovery process must be presented. The schema above is presenting the initial state of the recovery process.

### 4.3 Simulation Model

As mentioned in previous section, the recovery process contains three actions, diagnose, analyze, and allocation. The recovery process is connected to the tables of resources i.e. master allocation table, critical service tables and non-critical service tables. Based on those tables, the process will check the availability of possible resource for reconfiguration purposes. The simulation model is shown in Figure 4-1. This simulation is created merely to simulate how the proposed model works.



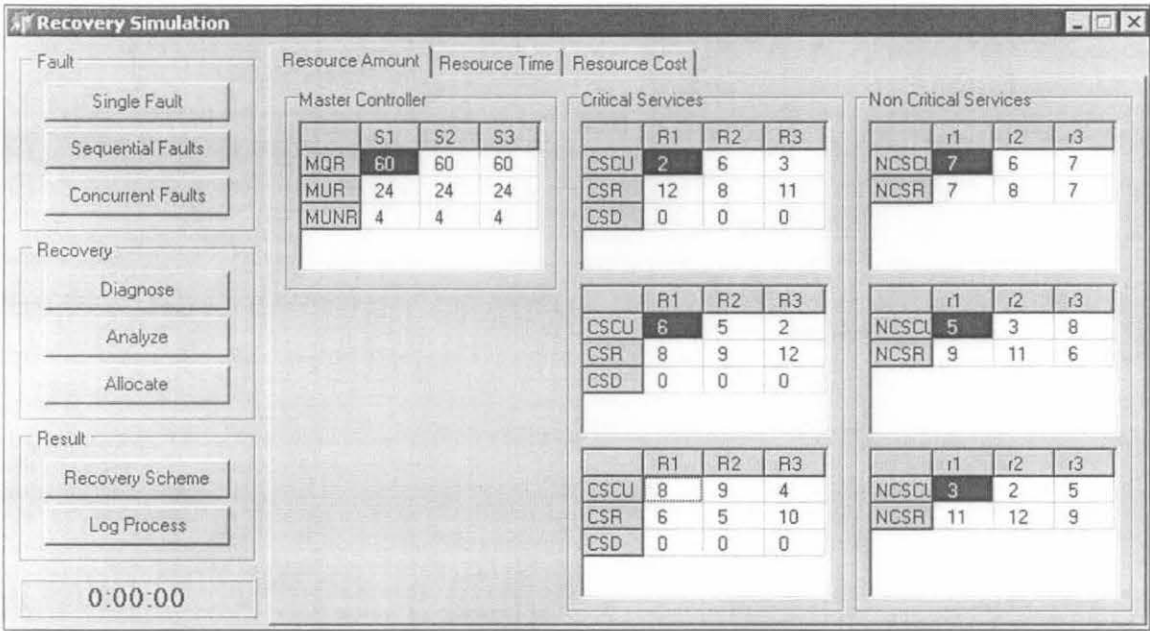


Figure 4-1 : Resource Quantity User Interface

The simulation contains three action panels and three data pages. The data page contains the resource quantity, resource response time and resource cost respectively. On the resource quantity page, the entire resource quantity follows the rules of Resource Balancing. The page contains one grid for the master controller table, three grids for critical service table and three grids for non critical service table. All of the value in the grids can be changed manually. The fault panel contains three buttons for simulating the fault. Single fault button will trigger damage in a critical service grid. Sequential faults button triggers damages in two critical service grids. Concurrent faults button triggers damages in all critical service grids. The recovery actions buttons are placed in the next panel. Each button represents one recovery action. When the button is pressed, a pop up form will appear to show the result of each process. The result panel is to show the recovery scheme found and the log on entire process.

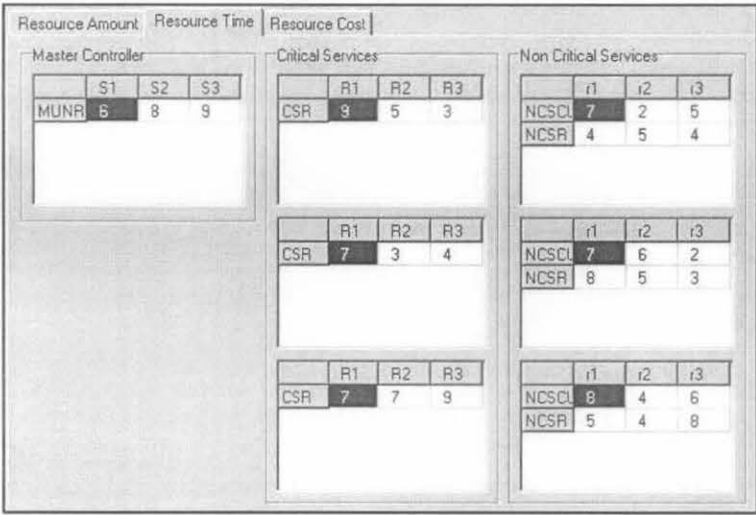


Figure 4-2 : Resource response time User Interface

Figure 4-2 shows the response time of each occupied resource in the reconfiguration. User may input each value manually based on the data. The response time uses 'unit time' as its unit.

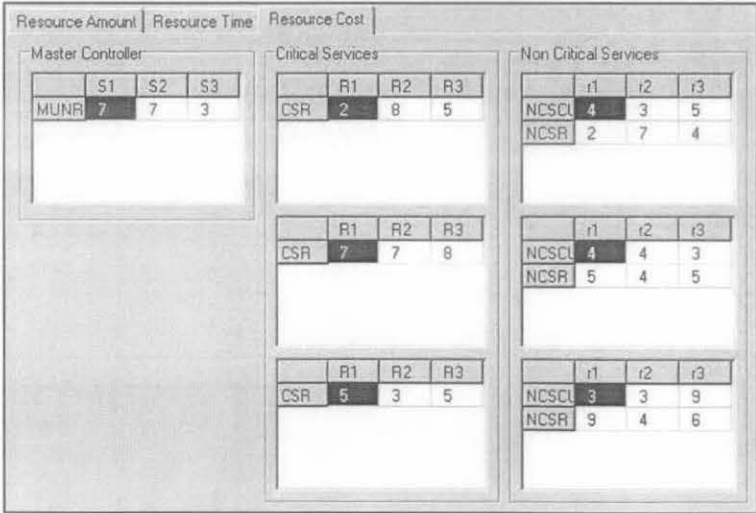


Figure 4-3 : Resource cost User Interface

Figure 4-3 shows the resource cost of each involved resource in the reconfiguration. Each value in every cell is manually inputted like the response time. The resource cost is given 'unit cost' as its unit.

4.3.1 Single Fault Case Study

To test the first type of fault, this work uses one case study as shown graphically in Figure 4-4, the delivery unit system of post office. There is a central post office in a town and some branch offices in the suburban area. Each post office has a unit called delivery unit with its mail-man. Let say a group of mail-man is a resource, some suburban post offices are set as critical services, some other suburban post offices are set non-critical service and the central post office in the town is the master resource controller.

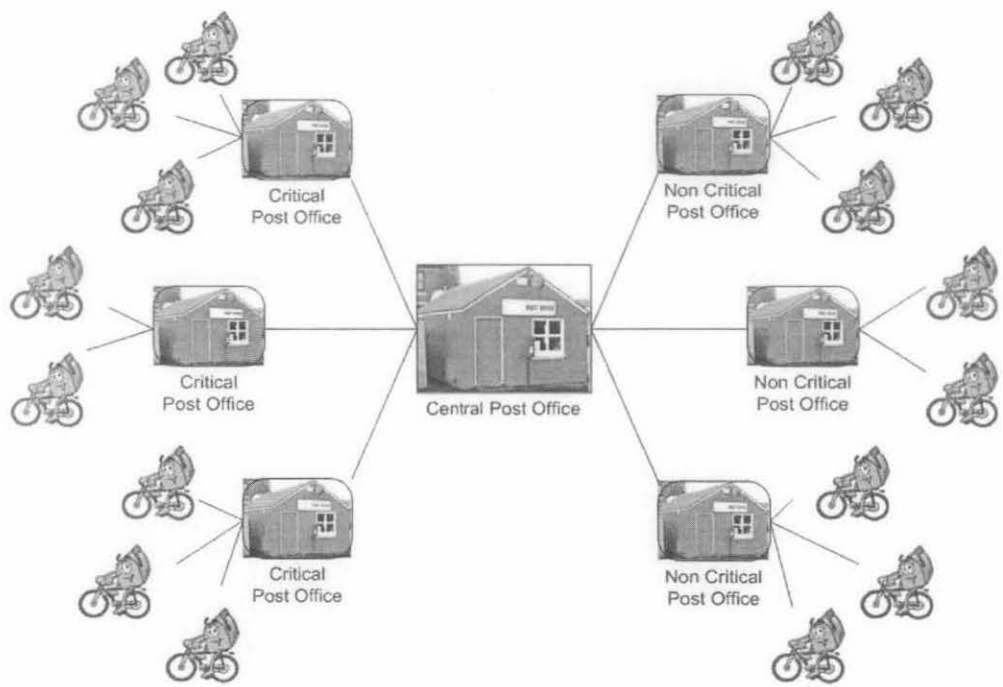


Figure 4-4 : Post Office Mail-Men Case Study

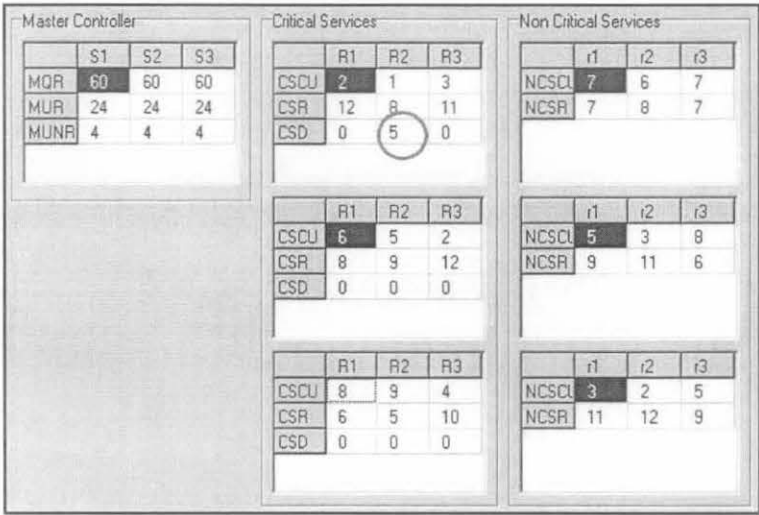


Figure 4-5 : Damage on one Critical Post Office

Figure 4-5 shows some critical suburban post office resources of  $R_2$  being damaged, called  $CSD_{12} = 5$ , highlighted in circle. To fix the error, scenario 1 will be executed and check the office's redundant unit,  $CSR_{12} = 8$ . Since it is enough, it is possible to execute the resource re-allocation process. Hence, to reconfigure 5 delivery units of  $CSD_{12}$ , the recovery process needs to allocate 5 units from  $CSR_{12}$ , see Figure 4-6.

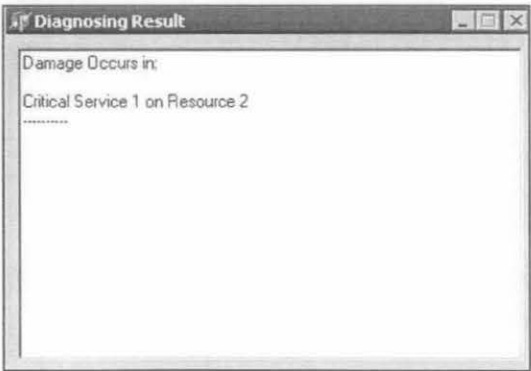


Figure 4-6 : Diagnose Result for Single Fault

The diagnose process (Figure 4-6) is running to set where the damage has taken place. It shows  $CSD_{12}$ , which means the first critical suburban post office at mail-man group 2.



Figure 4-7 : Analysis Result for Single Fault

Figure 4-7 shows the analysis outcome based on diagnosed parameters. It says that 5 units of mail-man of group 2 are in faulted,  $CSD_{12}(\alpha) = 5$ .

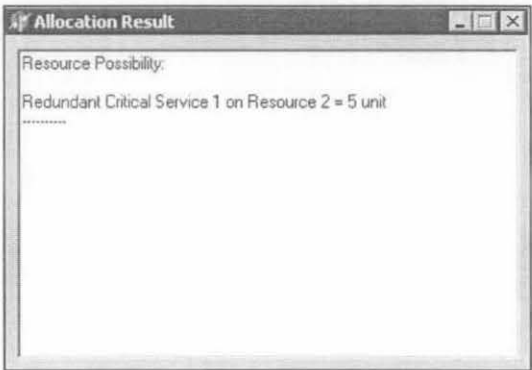


Figure 4-8 : Allocation Result for Single Fault

Figure 4-8 shows the allocation process, the last part of recovery process. Since  $CSR_{12}(\alpha) \geq CSD_{12}(\alpha)$ , then allocate 5 units from  $CSR_{12}$ , the redundant delivery units of first critical suburban post office.

Figure 4-9 shows the recovery scheme, final response time, and final cost as the result of the recovery process for single fault.  $\delta : CSR_{12}$ ,  $T(\delta) = 5$  units time,  $C(\delta) = 5 \times 8 = 40$  unit cost, and  $N(\delta) = 1$ .

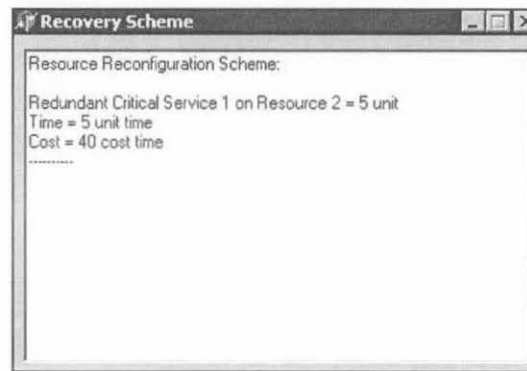


Figure 4-9 : Feasible Recovery Scheme for Single Fault

### 4.3.2 Multiple Sequential Faults Case Study

If another error occurred to one other group of mail-man of the critical post office after the first critical service being in fault, it means sequential faults has occurred.

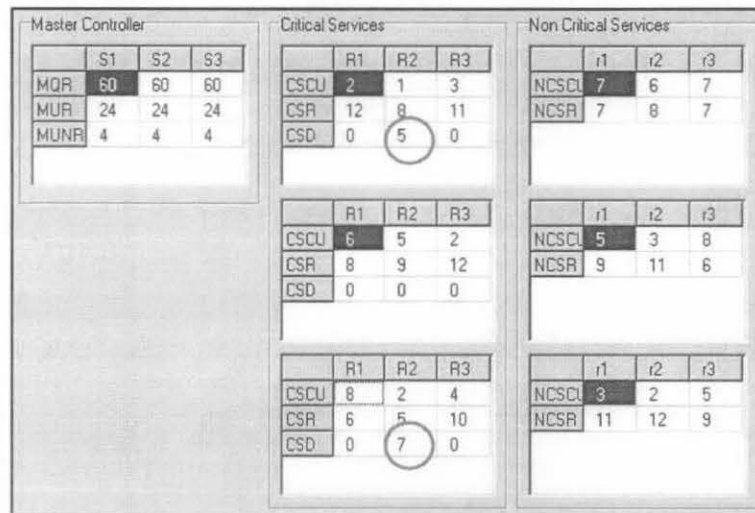


Figure 4-10 : Multiple Sequential Faults Case

Figure 4-10 shows another urban critical post office (CS3) is in fault after the first critical post office (CS1), highlighted in 2<sup>nd</sup> red circle. 7 mail-man units of group 2,  $CSD_{32}$  are damaged after 5 mail-man units of group 2 of CS1.



Figure 4-11 : Sequential Faults Diagnosed Result

Figure 4-11 shows the diagnosed results of both errors. It shows where they have taken place. The errors occurred in  $CSD_{12}$  and  $CSD_{32}$ .

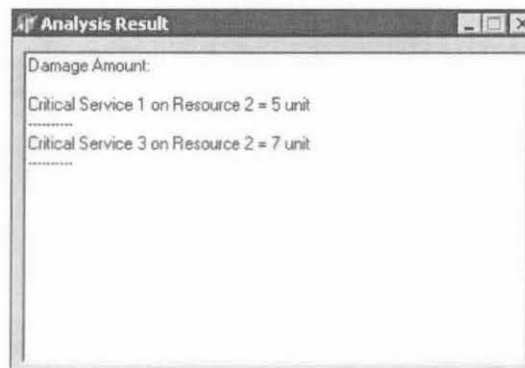


Figure 4-12 : Analysed Results for Sequential Faults

Figure 4-12 shows the result of analysis part of the engine for this case. It shows 5 main-man units of group 2 of critical post office 1 and 7 mail-man units of group 2 of critical post office 2 are being faulted.  $CSD_{12}(\alpha) = 5$  units and  $CSD_{32}(\alpha) = 7$  units.

For  $CSD_{12}(\alpha) = 5$ , it will check the redundant delivery units from its service. Since  $CSR_{12}(\alpha) \geq CSD_{12}(\alpha)$ , then allocate 5 units from  $CSR_{12}$ . The same action goes to the



next errors;  $CSD_{32}(\alpha) = 7$ . Since  $CSR_{32}(\alpha) < CSD_{32}(\alpha)$ , the scenario 2 will be activated. The idle mail-man of the central post office will be checked.  $MUNR_2(\gamma) = 4$  units. It means, to fix the remaining 2 units of damaged resources; the analysis process may take from the central post office idle resources. Consequently, the balance of  $MUNR_2(\gamma)$  will be 2 resources.

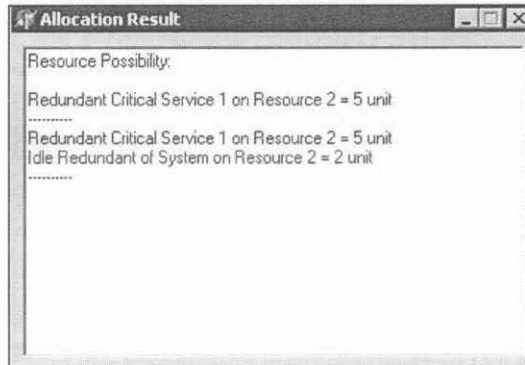


Figure 4-13 : Allocation result for sequential faults

The allocation process results are shown in Figure 4-13. It shows that to fix 7 damaged resources of the second error, 5 units of mail-man will be taken from the redundant resources of its service and 2 units of mail-man taken from idle redundant of central post office. There is no other feasible scheme of this allocation.

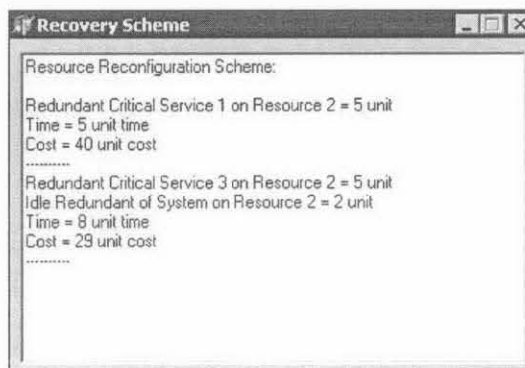
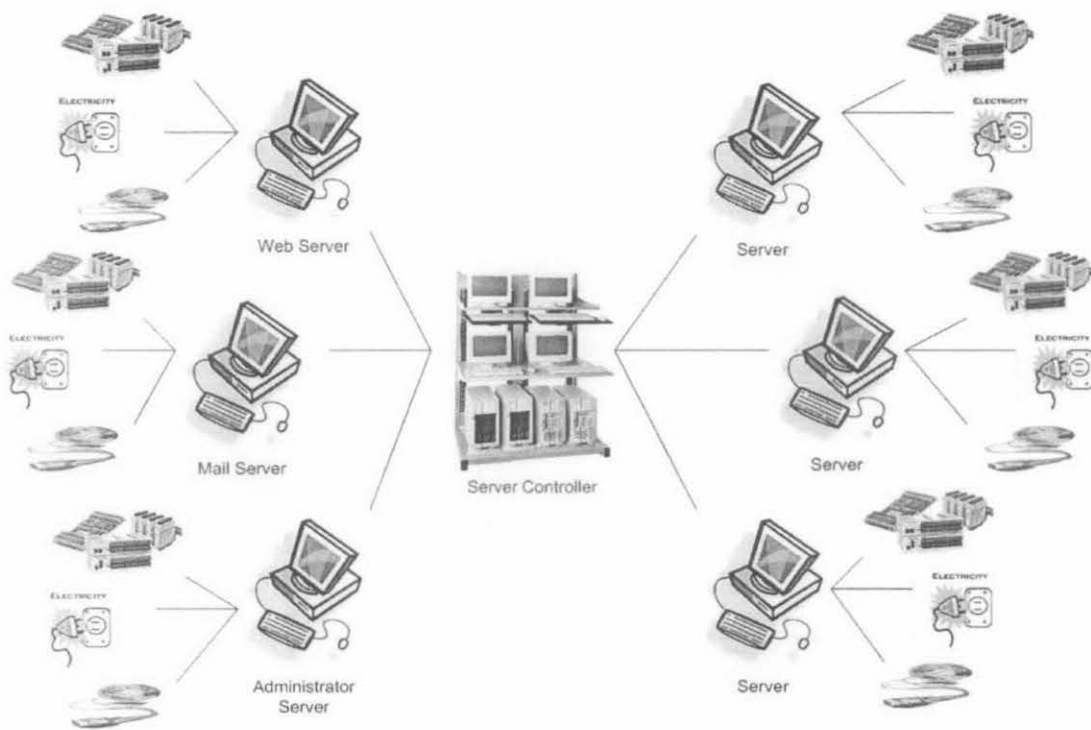


Figure 4-14 : Feasible Scheme for sequential faults

Finally, in Figure 4-14, the scheme of reconfiguration provided by this engine is displayed. For the first error  $\delta_1 = \{CSR_{12}\}$ ,  $T(\delta_1) = 5$ , and  $C(\delta_1) = 40$ . For the next error  $\delta_2 = \{CSR_{32}, MUNR_2\}$ ,  $T(\delta_2) = 8$ , and  $C(\delta_2) = 29$ .

### 4.3.3 Multiple Concurrent Faults Case Study

For multiple concurrent faults type, let's take a computer server system network of a company. The company has three critical servers supporting the service for its customers; web server, mail server, and administration server. To support the server process, every server needs some resources, such as connection, electric power, and its computer peripherals. Some non-critical servers also presented for other services. Figure 4-15 depicts this scenario.



**Figure 4-15 : Computer Server Systems Network**

Let the web server, mail server, and administration server is the  $CS_1, CS_2$  and  $CS_3$  respectively. The level of web server = 1, mail server = 2, and administration server = 3.

Level 1 is more important than level 3. The connection, electric power, and computer peripheral are the resources. The other servers are the non critical services ( $NCS_1, NCS_2, NCS_3$ ).

For example, multiple concurrent faults occurred to all the CS. The faults destroyed the electric power of all critical servers. If the three reconfiguration requests arrive in the same time and the critical service level is  $l_i : \{1, 2, 3\}$ , the queue will be  $Q : \{(CSD_{web\ server}), (CSD_{mail\ server}), (CSD_{ad\ min\ istration\ server})\}$ .

Resource Amount				Resource Time				Resource Cost			
Master Controller				Critical Services				Non Critical Services			
	S1	S2	S3		R1	R2	R3		r1	r2	r3
MQR	120	120	120	CSCU	16	17	10	NCSCU	17	15	12
MUR	24	24	24	CSR	8	7	14	NCSR	7	9	12
MUNR	4	4	4	CSD	0	0	0				
					R1	R2	R3		r1	r2	r3
				CSCU	7	18	11	NCSCU	13	13	16
				CSR	17	6	13	NCSR	11	11	8
				CSD	0	0	0				
					R1	R2	R3		r1	r2	r3
				CSCU	12	13	15	NCSCU	17	15	14
				CSR	12	11	9	NCSR	7	9	10
				CSD	0	0	0				

Figure 4-16 : Initial State of the network

Figure 4-16 shows the initial condition of the system at normal state. Each service has 20 units required resources and 4 units redundant for each resource. The master controller has 4 units idle/unused redundant for each resource. There is no damage happened in any service. The  $MQR(\gamma) = 120$  means the summary of all required resources in all services for one type of resource.





The diagnosed results can be seen in Figure 4-20 as the beginning process of recovery. The figure shows damages happen on  $CSD_{12}$ ,  $CSD_{22}$ , and  $CSD_{32}$ .

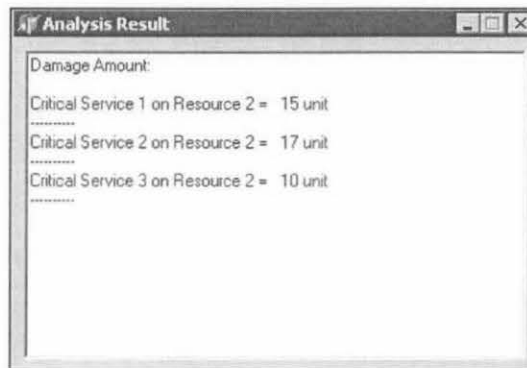


Figure 4-21 : Analysis result of Multiple Concurrent Faults

Figure 4-21 shows the damage amount resulted by analysis process. The amount is captured by tracing the place of damage identified in previous process.  $CSD_{12}(\alpha) = 15$ ,  $CSD_{22}(\alpha) = 17$ , and  $CSD_{32}(\alpha) = 10$ .

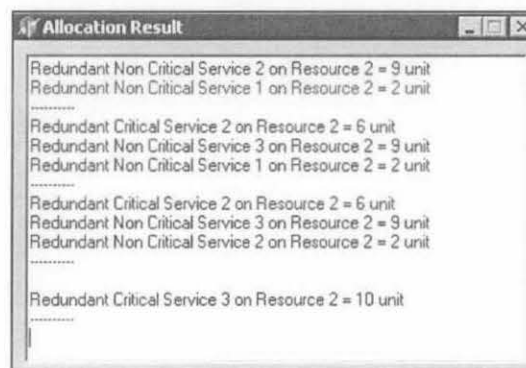


Figure 4-22 : Allocation result of Multiple Concurrent Faults

Figure 4-22 shows the allocation result of recovery process. The possible resources are chosen by the scenario. In this case, none of the non-critical service resources are being pre-

empted for the reconfiguration purpose. The detail process is shown in Table 4-1 : Allocation Process for Multiple Concurrent Faults. Rows with grey colour are the solution based on hierarchical decision.



Table 4-1 : Allocation Process for Multiple Concurrent Faults

Damage	Feasible Schemes	Response Time	Cost	Resource Number
$CSD_{12}(15)$	$CSR_{12}(7), MUNR_2(4), NCSR_{12}(4)$	$\max(9,7,7)=9$	$7*6 + 4*9 + 4*7 = 106$	3
	$CSR_{12}(7), MUNR_2(4), NCSR_{22}(4)$	$\max(9,7,6)=9$	$7*6 + 4*9 + 4*2 = 86$	3
	$CSR_{12}(7), MUNR_2(4), NCSR_{32}(4)$	$\max(9,7,5)=9$	$7*6 + 4*9 + 4*8 = 110$	3
$CSD_{22}(17)$	$CSR_{22}(6), NCSR_{12}(9), NCSR_{22}(2)$	$\max(5,7,6)=7$	$6*7 + 9*7 + 2*2 = 109$	3
	$CSR_{22}(6), NCSR_{12}(9), NCSR_{32}(2)$	$\max(5,7,5)=7$	$6*7 + 9*7 + 2*8 = 121$	3
	$CSR_{22}(6), NCSR_{22}(9), NCSR_{32}(2)$	$\max(5,6,5)=6$	$6*7 + 9*2 + 2*8 = 76$	3
	$CSR_{22}(6), NCSR_{22}(9), NCSR_{12}(2)$	$\max(5,6,7)=7$	$6*7 + 9*2 + 2*7 = 74$	3
	$CSR_{22}(6), NCSR_{32}(9), NCSR_{12}(2)$	$\max(5,5,7)=7$	$6*7 + 9*8 + 2*7 = 128$	3
	$CSR_{22}(6), NCSR_{32}(9), NCSR_{22}(2)$	$\max(5,5,6)=6$	$6*7 + 9*8 + 2*2 = 118$	3
$CSD_{32}(10)$	$CSR_{32}(10)$	3	$10*2 = 20$	1

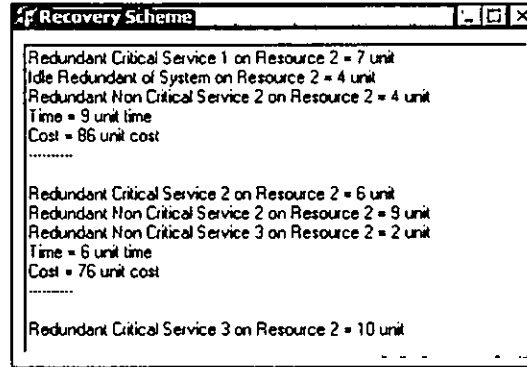


Figure 4-23 : Recovery Schemes of Multiple Concurrent Faults

The last result of recovery process is shown in Figure 4-23. The reconfigurations based on feasible schemes are:

$$\delta_1 : \{CSR_{12}, MUNR_2, NCSR_{22}\}, T(\delta_1) = 9 \text{ unit time}, C(\delta_1) = 86 \text{ unit cost}, N(\delta_1) = 3$$

$$\delta_2 : \{CSR_{22}, NCSR_{22}, NCSR_{32}\}, T(\delta_2) = 6 \text{ unit time}, C(\delta_2) = 76 \text{ unit cost}, N(\delta_2) = 3$$

$$\delta_3 : \{CSR_{32}\}, T(\delta_3) = 3 \text{ unit time}, C(\delta_3) = 20 \text{ unit cost}, N(\delta_3) = 1$$

#### 4.4 Result and Analysis

The interest of this research is to find a technique how to recover a damage service by reconfiguring its resources with minimum service disruption. There are some analysis of the model based on the simulation result and comparison to the closest work related to that interest:

1. Based on the case study simulation, it is shown that the model is not limited to computer system, but also any system, as long as the system has the same architecture as mentioned in Figure 3.1. The model can solve multiple faults that happened to the same resource.

2. In Table 4-1, the hierarchical order based on requisite conditions is shown when the reconfiguration scheme has to be chosen. It also shows that if the master idle resource has already booked, it will not book by other request.
3. Compared to ERAS [Wang, 2006], the closest work to this research, the simulation shows that the developed model is capable of making a reconfiguration with minimal affection to the currently used (pre-empted) resources of any services. While ERAS can only reconfigure from pre-empted resources of non-critical service, this model has an advantage over ERAS in giving a more generic solution.
4. The developed model can also handle three types of fault which is the common faults happen in the real situation. The priority queue makes every faults condition is possible to be solved.

#### 4.5 Summary

This chapter has shown how the developed model works by specifying it using Z specification language and simulate it with some case studies. By specifying the model into formal model, it helps the other developers to understand the general logic of the process. The Z specification language has been proved using Z/EVES simulator. In the simulation part, the time efficiency of reconfiguration can be shown by minimally affecting another service. The model is very applicable to any system, not only computer system. The simulation can be called as a generic simulation because it demonstrates the model in every possible case.

## CHAPTER 5 : CONCLUSION

The research on survivability has become an interesting topic in the field of recovery, and one of its emphases is on how to improve the system abilities for damage recovery as conditions permit. In this research, a basis recovery model for survivable critical service based on resource reconfiguration has been developed. The model encompasses several detailed scenarios of resource re-allocation in reconfiguring the damaged system. The model assigned available redundant resources in the system to the critical services and pre-empting the currently used resources from non-critical services. This reconfiguration process in the developed model is a new approach in fault tolerance perspective.

Based on the requisite conditions; resource response time, resource cost, and number of resources (especially the pre-emptive resources), the model has shown that is it very effective useful model. The model also addressed several fault types, namely single fault, multiple sequential faults, and multiple concurrent faults that may happen to the critical services.

This work has defined the concept of 'resource balancing' and found it very useful to ensure the resource reconfiguration performs properly under various scenarios.

This work also proposed a system architecture where the system can survive with the model developed. The system contains critical and non critical services also master resource controller. The system administrator will be the person in charge who set the system as the proposed architecture.

Regarding to the Z formal specification which is created using Z/EVES, it is shown that formal specification can help in pre-implementation such model in any programming development. The proven logic of the specification helps in describing the model to programmers or common people.

The model developed here can help the system administrators make better decisions to enhance the survivability of the system. With real data, administrator should be able to achieve decisions closer to the optimal survivability.

## CHAPTER 6 : FUTURE WORK

There are several directions for future work. First, it is important to create the application of this model and implement it into a real-time system. Several case studies need to be considered, including the use of an agent application for each system. In real-time system, optimization mechanism could be used in the model. Redundancy optimization algorithms are being tested for efficiencies computability to determine its computational complexity. The optimization must include the three requisite conditions, specially the cost factor.

A decision support system should also be built in the further research. The need of improved DSS based on requisite conditions and other factors such as Multi Criteria Decision Making (MCDM) becomes important to reconfigure a damage resource accurately.

With regards to many real-time systems that commonly include fail states; the success probability of recovery must be considered. Other quantification factors such as graceful degradation time limit and resource type can be taken into account to enhance the model.

There are other redundancy mechanisms such as data/information redundancy, time redundancy, method redundancy and software redundancy that may be used to extend this work into a more advanced model.

Some assumptions have been made in the development of the model; such as establishment of checkpoint, damage assessment, synchronization, and resource restoration are factors that are considered to have been taken care of by other mechanisms. They can be realised in future work.

The general interconnected network recovery model is developed merely for generalizing the recovery within a network. The specification and simulation part of that model should be built in future work.

In autonomic computing, the next step of recovery in survivable system is self-healing system [Koopman, 2003]. This research is in fact one aspect of self-healing mechanisms that invokes recovery with minimal human intervention. There are many aspects that might be integrated into this model to develop a full scale self-healing system.

## REFERENCES

- [Ateniese, 2006] G. Ateniese et al., "Survivable Monitoring in Dynamic Networks", *IEEE Transactions on Mobile Computing*, Volume 5-9, September 2006.
- [Aung, 2006] K.M.M. Aung et al., "Survival of the Internet Application: A Cluster Recovery Model", *Proceeding of the 6th IEEE International Symposium on Cluster Computing and the grid Workshop*, 2006.
- [Bohannon, 2003] P. Bohannon et al., "Detection and Recovery Techniques for Database Corruption", *IEEE Transactions on Knowledge and Data Engineering*, Volume 15-5, September/October 2003.
- [Caldera, 2000] Jose Caldera, "Survivability Requirements for the U.S. Health Care Industry", Thesis submitted to the Information Networking Institute, Carnegie Mellon University, May 2000.
- [Desmedt, 2002] Y. Desmedt, "Is there a Need for Survivable Computation in Critical Infrastructures?", *Information Security Technical Report*, Vol 7/2, 2002.
- [Ellison, 1999a] R. Ellison et al., "Survivable network systems: An emerging discipline". *Technical Report*, CMU/SEI-97-153, 1997. Revised 1999.
- [Ellison, 1999b] R.J. Ellison et al., "Survivability: Protecting your Critical Systems", CMU/SEI, IEEE, 1999.
- [Ellison, 1999c] R.J. Ellison et. al., "An Approach to Survivable System", Carnegie Mellon University, 1999.
- [Elnozahy, 2004] E.N. Elnozahy and J.S. Plank, "Checkpointing for Peta-Scale System: A Look into Future of Practical Rollback-Recovery", *IEEE Transactions on Dependable and Secure Computing*, Volume 1-2, April-June 2004.



- [Florio, 2001] V. De Florio et al., "The Recovery Language Approach for Software-Implemented Fault Tolerance", *Proceedings of 9<sup>th</sup> Euro Micro Workshop on Parallel and Distributed Processing*, IEEE, 2001.
- [Fung, 2005] C. Fung and P.C.K. Hung, "System Recovery through Dynamic Regeneration of Workflow Specification", *Proceedings of the eighth IEEE ISORC*, 2005.
- [Ghiasi, 2004] S. Ghiasi et. al., "An Optimal Algorithm for Minimizing Run-Time Reconfiguration Delay", *ACM Transaction on Embedded Computing Systems*, Vol. 3/2, May 2004.
- [Greenwood, 2005] G.W. Greenwood, "On The Practically of Using Intrinsic Reconfiguratin for Fault Recovery", *IEEE Transactions on Evolutionary Computation*, Volume 9-4, August 2005.
- [Herzberg, 2004] Meir Herzberg, "The Cycle-Oriented Approach: A Pragmatic Concept for Resource Allocation in Multi-Service Survivable Networks", *Proceedings of 11<sup>th</sup> Telecommunication Network Strategy and Planning Symposium*, IEEE, 2004.
- [Hiltunen, 2001] M.A. Hiltunen et. al., "Enhancing Survivability of Security Services using Redundancy", IEEE, 2001.
- [Hiltunen, 2003] Matti A. Hiltunen et al., "Building Survivable Services Using Redundancy and Adaptation", *IEEE Transactions on Computer*, Volume 52-2, February 2003.
- [Ho, 2007] K.S. Ho and K.W. Cheung, "Generelized Survivable Network", *IEEE/ACM Transactions on Networking*, Volume 15-4, August 2007.
- [Jann, 2003] J. Jann et. al., "Dynamic Reconfiguration: Basic building blocks for autonomic computing on IBM pSeries servers", *IBM System Journal* Vol. 42/1, 2003.
- [Janota, 2000] Ales Janota, "Using Z Specification for railway interlocking safety", *PERIODICA POLYTECHNICA SER. TRANSP. ENG.* Vol. 28/1-2, 2000.

- [Jha, 2000] S. Jha et. al., "Survivability Analysis of Network Specifications", IEEE, 2000.
- [Jha, 2001] S. Jha and J.M. Wing, "Survivability Analysis of Networked Systems", *Proceeding of ICSE 2001*, Toronto, 2001.
- [Knight, 1998] J. Knight et al., "Error Recovery in Critical Infrastructure Systems", *Proceedings of the 1998 Computer Security, Dependability, and Assurance (CSDA'98) Workshop*, Williamsburg, VA, November 1998.
- [Knight, 2000] J.C. Knight et. al., "Survivability Architectures: Issues and Approaches", *Proceedings of DISCEX'00*, IEEE, 2000.
- [Knight, 2003] J.C. Knight et. al., "Towards a rigorous definition of information system survivability", *DARPA Information Survivability Conference and Exposition*, IEEE, 2003.
- [Koopman, 2003] P. Koopman, "Element of the Self-Healing System Problem Space", *Proceedings of the ICSE WAD03*, 2003.
- [Koroma, 2003] J. Koroma et al., "A Generalized Model for Network Survivability", *Proceeding of TAPIA 2003*, ACM, Atlanta, 2003.
- [Lewandowski, 2001] S.M. Lewandowski et al., "SARA: Survivable Autonomic Response Architecture", *Proceedings of DISCEX*, IEEE, 2001.
- [Lin, 2005] X. Lin et al., "A Framework for Quantifying Information System Survivability", *Proceeding of ICITA 2005*, IEEE, 2005.
- [Linger, 2001] R.C. Linger and A.P Moore, "Foundations for Survivable System Development: Service Traces, Intrusion Traces, and Evaluation Models", *Technical Report CMU/SEI-2001-TR-029*, October 2001.
- [Linger, 2002] R.C. Linger et. al., "Life-Cycle Models for Survivable Systems", *Technical Report CMU/SEI-2002-TR-026*, 2002.
- [Mead, 2000] N.R. Mead et. al., "Survivability Network Analysis Method", *Technical Report*, CMU/SEI-2000-TR-013, Carnegie Mellon University, 2000.

- [Mead, 2003] N.R. Mead, "Requirements Engineering for Survivable Systems", *Technical Note CMU/SEI-2003-TN-013*, September 2003.
- [McDermott, 2005] J. McDermott, "Attack-Potential-Based Survivability Modelling for High-Consequence Systems", *Proceeding of the 3rd International Workshop on Information Assurance 2005*, IEEE, 2005.
- [Melhem, 2004] R. Melhem et al., "The Interplay of Power Management and Fault Recovery in Real-Time Systems", *IEEE Transactions on Computers*, Volume 53-2, February 2004.
- [Moitra, 2000] S.D. Moitra and S.L. Konda, "a Simulation Model for Managing Survivability of Networked Information System", *Technical Report*, CMU/SEI-2000-TR-020, 2000.
- [Mosse, 2003] Daniel Mosse et al., "A Nonpreemptive Real-Time Scheduler with Recovery from Transient Faults and Its Implementation", *IEEE Transactions on Software Engineering*, Volume 29-8, August 2003.
- [Müller, 2006] H.A. Müller et. al., "Autonomic Computing", Technical Note CMU/SEI-2006-TN-006, April 2006.
- [Park, 2004] J. Park and P. Chandramohan, "Static vs. Dynamic Recovery Models for Survivable Distributed Systems", *Proceeding of the 37<sup>th</sup> Hawaii Internal Conference on Systems Sciences (HICSS'04)*, IEEE, 2004.
- [Park, 2005] B. Park et al., "A Self-healing Mechanism for an Intrusion Tolerance System", *TrustBus 2005*, Copenhagen, Springer, 2005.
- [Pazel, 2002] D.P. Pazel et. al., "Neptune: A Dynamic Resource Allocation and Planning System for a Cluster Computing Utility", *Proceedings of CCGRID'02*, IEEE, 2002.
- [Popstojanova, 2001] K. Goseva-Popstojanova et al., "Characterizing Intrusion Tolerance Systems using State Transition Model", *DARPA Information Survivability Conference and Exhibition*, June 2001.

- [Ravindran, 2002] B. Ravindran, "Engineering Dynamic Real-Time Distributed Systems: Architecture, System Description Language, and Middleware", *IEEE Transactions on Software Engineering*, Volume 28-1, 2002.
- [Rose, 2003] Scott Rose et. al., "Improving Failure Responsiveness in Jini Leasing", *Proceedings of DISCEX'03*, IEEE, 2003.
- [Shao, 2005] B.B.M. Shao, "Optimal Redundancy Allocation for Information Technology Disaster Recovery in the Network Economy", *IEEE Transactions on Dependable and Secure Computing*, Vol. 2/3, July-September 2005.
- [Sullivan, 1999] K. Sullivan et. al., "Information Survivability Control Systems", *Proceedings of International Conference of Software Engineering*, IEEE, 1999.
- [Tarvainen, 2004] P. Tarvainen, "Survey of the Survivability of IT Systems", *The 9<sup>th</sup> Nordic Workshop on Secure IT-Systems*, Finland, November 2004.
- [Taylor, 2002] C. Taylor and J. Alves-Foss, "Attack Recognition for System Survivability: A Low-level Approach", *Proceedings of the 37<sup>th</sup> HICSS*, 2002.
- [Tirtea, 2006] R. Tirtea et al., "Fault Tolerance Adaptation Requirement vs. QoS, Real-Time and Security in Dynamic Distributed Systems", *Proceedings of RAMS*, IEEE, 2006.
- [Wang, 2003] F. Wang et al., "Analysis of Techniques for Building Intrusion Tolerant Server Systems", *Proceedings of MILCOM*, IEEE, 2003.
- [Wang, 2006] J. Wang et al., "ERAS – an Emergence Response Algorithm for Survivability of Critical Services", *Proceeding of IMSCCS 2006*, IEEE, 2006.
- [Wells, 2000] D. Wells et al., "Software Survivability", *Proceedings of DISCEX'00*, IEEE, 2000.
- [Westmark, 2004] V.R. Westmark, "A Definition for Information System Survivability", *Proceedings of the 37<sup>th</sup> HICSS'04*, IEEE, 2004.

- [Woodcock, 1996] J.C.P. Woodcock and J Davies, “ Using Z: Specification, Refinement, Proof”. *Prentice Hall International Series in Computer Science*, 1996.
- [Zhao, 2006] G. Zhao et al., “A Novel Quantitative Analysis method for Network Survivability”, *Proceeding of IMSCCS 2006*, IEEE, 2006.

## APPENDIX A: The Allocation Flowchart

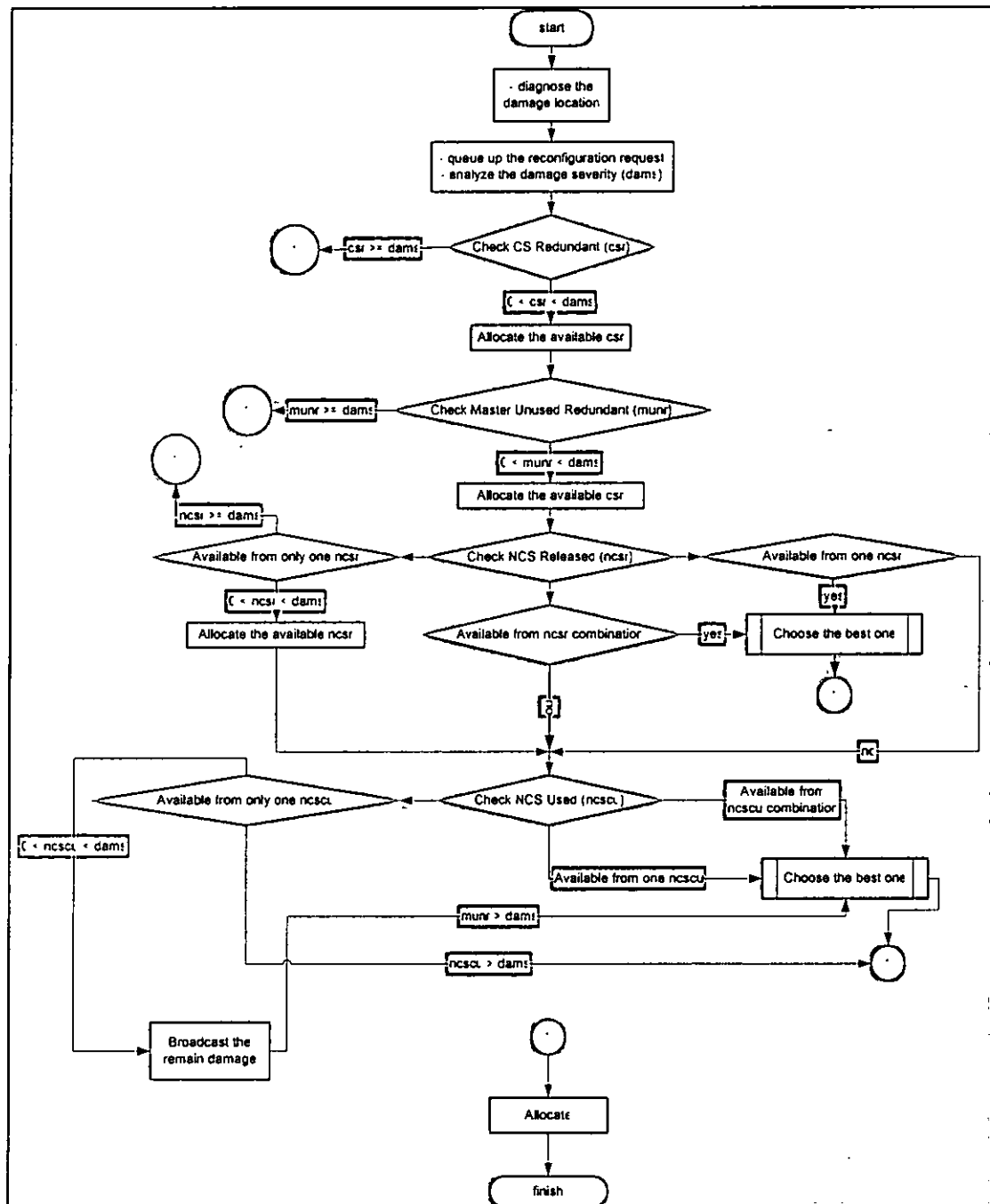


Figure A-1: Allocation Flowchart

## APPENDIX B: Simulation Program Code

The simulation contains two main processes: the initial process and the recovery process. Figure B-1 illustrates the initial condition for the simulation. Meanwhile, Figure B-2 illustrates the recovery process.

```
// unit declaration
Uses
Windows, Messages, SysUtils, Variants, Classes, Graphics,
Controls, Forms, Dialogs, Grids, StdCtrls, ComCtrls, ExtCtrls,
Math;

// grid naming
procedure naming;
begin
  with {grid_name} do
  begin
    // name the grid;
  end;
end;

// grid filling
procedure filling;
var
  i : integer;
begin
  randomize;
  with {grid service} do
  begin
    for i:=1 to 3 do
      // fill the grid
    end;
  end;
end;
```

Figure B-1: The Initial Condition Process

```
// Main Procedure
procedure recovery;
begin
// define all the needed variable
// while remain_damage <> 0 and result not found
if {csr >= damage} then
begin
    result := damage;
    num[x] := 1;
    time[t] := csr_time;
    cost[c] := csr_cost * result;
end else
if {csr < damage} then
begin
    result := csr;
    remain_damage := damage - csr;
    num[x] := 1;
    time[t] := csr_time;
    cost[c] := csr_cost * result;
    damage := remain_damage;
    if {mstr >= damage} then
        begin
            result := result+damage;
            mstr := mstr - damage;
            num[x] := num[x] + 1;
            time[t] := max(time[t],mstr_time);
            cost[c] := cost[c]+(mstr_cost*result);
        end else
        if {mstr < damage} then
            begin
                result := result+mstr;
                remain_damage := damage - mstr;
                num[x] := num[x] + 1;
                time[t] := max(time[t],mstr_time);
                cost[c] := cost[c]+(mstr_cost*result);
                damage := remain_damage;
```



```

if {ncsr[m] >= damage} then
  begin
    result := result+damage;
    num[x] := num[x] + {count(ncsr[m])};
    time[t] := max(time[t],ncsr_time[m]);
    cost[c] := cost[c]+(damage * ncsr_cost[m]);
  end else
  if {ncsr[m] < damage} then
    begin
      result := result+ncsr[m];
      remain_damage := damage - result;
      num[x] := num[x] + {count(ncsr[m])};
      time[t] := (max(time[t],ncsr_time[m]));
      cost[c] := cost[c]+(damage * ncsr_cost[m]);
      damage := remain_damage;
      if {ncscu[n] >= damage} then
        begin
          result := result+damage;
          num[x] := num[x] + {count(ncscu[n])};
          time[t] := max(time[t],ncscu_time[n]);
          cost[c] := cost[c]+(damage * ncscu_cost[n]);
        end else
        if {ncscu[n] < damage} then
          begin
            result := result+ncscu[n];
            remain_damage := damage - result;
            num[x] := num[x] + {count(ncscu[n])};
            time[t] := max(time[t],ncscu_time[n]);
            cost[c] := cost[c]+(damage * ncscu_cost[n]);
            damage := remain_damage;
          end;
        end;

      // choose the minimum {num[x] and time[t] and cost[c]}
      // choose the best scheme for reconfiguration
      // result found

    end.

```

Figure B-2: The Recovery Process